

where

$$F^* \simeq \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}. \quad (11.50)$$

The rectifying homographies are not unique. Two instances of rectification are shown in Figure 11.12. The interesting question is which of the many possible rectifications is the best, which we will discuss shortly.

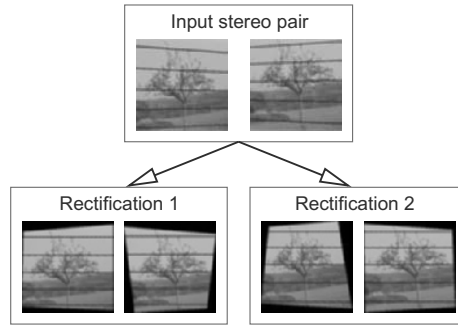


Figure 11.12: Two instances of many possible rectifications. *Courtesy of R. Šára and M. Matoušek, Czech Technical University, Prague.*

Algorithm 11.1: Image rectification

1. Epipoles are translated to infinity in both images.

Let $\mathbf{e}_L = [e_1, e_2, 1]^\top$ be the epipole in the left image and $e_1^2 + e_2^2 \neq 0$. This epipole is mapped to $\mathbf{e}^* \simeq [1, 0, 0]^\top$ as the rotation of the epipole $\mathbf{e}_L$ to the axis u and the projection

$$\hat{H}_L \simeq \begin{bmatrix} e_1 & e_2 & 0 \\ -e_2 & e_1 & 0 \\ -e_1 & -e_2 & e_1^2 + e_2^2 \end{bmatrix}. \quad (11.51)$$

2. Epipolar lines are unified to get a pair of elementary rectifying homographies.

Since $\mathbf{e}_R^* = [1, 0, 0]^\top$ is both left and right null space of $\hat{F}$, the modified fundamental matrix becomes

$$\hat{F} \simeq \begin{bmatrix} 0 & 0 & 0 \\ 0 & \alpha & \beta \\ 0 & \gamma & \delta \end{bmatrix} \quad (11.52)$$

and elementary rectifying homographies $\bar{H}_L, \bar{H}_R$ are chosen to make $\alpha = \delta = 0$ and $\beta = -\gamma$.

$$\bar{H}_L = H_S \hat{H}_L, \quad \bar{H}_R = \hat{H}_R, \quad \text{where } H_S = \begin{bmatrix} \alpha\delta - \beta\gamma & 0 & 0 \\ 0 & -\gamma & -\delta \\ 0 & \alpha & \beta \end{bmatrix}. \quad (11.53)$$

Then

$$F^* = \left(\hat{H}_R \right)^{-\top} F \left(H_S \hat{H}_L \right)^{-1}. \quad (11.54)$$

3. A pair of optimal homographies is selected from the class preserving the fundamental matrix F^* .

Let $\bar{H}_L, \bar{H}_R$ be elementary rectifying homographies (or some other rectifying homographies). Homographies H_L, H_R are also rectifying homographies provided they obey equation $H_R F^* H_L^\top = \lambda F^*$, $\lambda \neq 0$, which guarantees that images are kept rectified.

The internal structure of H_L, H_R permits us to understand the meaning of free parameters in the class of rectifying homographies

$$H_L = \begin{bmatrix} l_1 & l_2 & l_3 \\ 0 & s & u_0 \\ 0 & q & 1 \end{bmatrix} \bar{H}_L, \quad H_R = \begin{bmatrix} r_1 & r_2 & r_3 \\ 0 & s & u_0 \\ 0 & q & 1 \end{bmatrix} \bar{H}_R, \quad (11.55)$$

where $s \neq 0$ is a common vertical scale; u_0 is a common vertical shift; l_1, r_1 are left and right skews; l_2, r_2 are left and right horizontal scales; l_3, r_3 are left and right horizontal shifts and q is common perspective distortion.

This third step is necessary because elementary homographies may yield severely distorted images.

The algorithms differ by the way free parameters are selected. One approach minimizes residual image distortion [Loop and Zhang, 1999; Gluckman and Nayar, 2001]. Another approach takes into account how much the underlying data change using frequency spectrum analysis and minimizing image information loss [Matoušek et al., 2004].

11.6 Three cameras and trifocal tensor

Section 11.5 was devoted to the matching constraints between two views which manifest in epipolar geometry. We saw in Section 11.4.3 that matching constraints exist also amongst three and four views. This section describes the one for three views. Its form is that a set of trilinear functions of image coordinates must vanish.

We follow the derivation of the trifocal tensor from [Hartley and Zisserman, 2003]. The constraint among three views receives its simplest form as a formula that computes line $\mathbf{l}$ in the first view from a given line $\mathbf{l}'$ in the second view and a given line $\mathbf{l}''$ in the third view. The geometrical meaning of this construction is simple: back project the lines $\mathbf{l}'$ and $\mathbf{l}''$ to scene planes, find a common scene line of these planes, and project this line into the first view (see Figure 11.13).

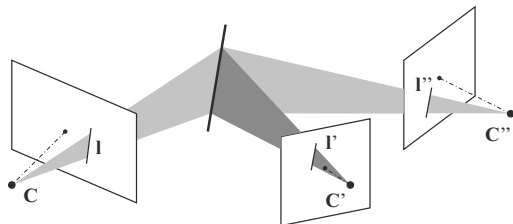


Figure 11.13: Illustration of the matching constraint among three views. Cameras have centers C, C', C'' and appropriate image planes. A line in 3D projects into lines $\mathbf{l}, \mathbf{l}', \mathbf{l}''$.
© Cengage Learning 2015.

Let the three views have camera matrices M , M' and M'' . Due to projective ambiguity described in Section 11.4.2, we can choose $M = [I \mid \mathbf{0}]$ without loss of generality. Then, using the result from expression (11.33), we have

$$M = [I \mid \mathbf{0}], \quad M' = [\tilde{M}' \mid \mathbf{e}'], \quad M'' = [\tilde{M}'' \mid \mathbf{e}''],$$

where the epipole $\mathbf{e}'$ and $\mathbf{e}''$ is the projection of the first camera center, $\mathbf{C} = [0, 0, 0, 1]^\top$, in the second and third camera, respectively.

To satisfy the constraint, the scene planes

$$\mathbf{a} = M^\top \mathbf{l} = \begin{bmatrix} \mathbf{l} \\ 0 \end{bmatrix}, \quad \mathbf{a}' = M'^\top \mathbf{l}' = \begin{bmatrix} \tilde{M}'^\top \mathbf{l}' \\ \mathbf{e}'^\top \mathbf{l}' \end{bmatrix}, \quad \mathbf{a}'' = M''^\top \mathbf{l}'' = \begin{bmatrix} \tilde{M}''^\top \mathbf{l}'' \\ \mathbf{e}''^\top \mathbf{l}'' \end{bmatrix}, \quad (11.56)$$

back-projected from the image lines as given by equation (11.21), have a scene line in common. This happens only if the vectors in equation (11.56) are linearly dependent, that is, $\mathbf{a} = \lambda' \mathbf{a}' + \lambda'' \mathbf{a}''$ for some scalars λ' and λ'' . Applying this to the fourth coordinates of the vectors in equation (11.56) yields $\lambda' \mathbf{e}'^\top \mathbf{l}' = -\lambda'' \mathbf{e}''^\top \mathbf{l}''$. Substituting to the first three coordinates of the vectors in equation (11.56) yields

$$\mathbf{l} \simeq (\mathbf{e}''^\top \mathbf{l}'') \tilde{M}'^\top \mathbf{l}' - (\mathbf{e}'^\top \mathbf{l}') \tilde{M}''^\top \mathbf{l}'' = (\mathbf{l}''^\top \mathbf{e}'') \tilde{M}'^\top \mathbf{l}' - (\mathbf{l}'^\top \mathbf{e}') \tilde{M}''^\top \mathbf{l}''.$$

The expression can be further re-arranged to

$$\mathbf{l} \simeq [\mathbf{l}'^\top T_1 \mathbf{l}'', \mathbf{l}'^\top T_3 \mathbf{l}'', \mathbf{l}'^\top T_3 \mathbf{l}'']^\top, \quad (11.57)$$

where we have denoted

$$T_i = \mathbf{m}'_i \mathbf{e}''^\top - \mathbf{m}''_i \mathbf{e}'^\top, \quad i = 1, 2, 3 \quad (11.58)$$

and $\tilde{M}' = [\mathbf{m}'_1 \mid \mathbf{m}'_2 \mid \mathbf{m}'_3]$, $\tilde{M}'' = [\mathbf{m}''_1 \mid \mathbf{m}''_2 \mid \mathbf{m}''_3]$. The three 3×3 matrices T_i can be seen as slices of the $3 \times 3 \times 3$ **trifocal tensor**.

Expression (11.57) is bilinear in the coordinates of image lines and describes how to compute the image line in the first view given lines in the other two views. In Section 11.4.3, we derived that there is a single trilinear function involving point $\mathbf{u}$ in the first image, line $\mathbf{l}'$ and line $\mathbf{l}''$ that vanishes if there exists a scene point projecting to these. It follows from the incidence relation $\mathbf{l}'^\top \mathbf{u} = 0$

$$[\mathbf{l}'^\top T_1 \mathbf{l}'', \mathbf{l}'^\top T_3 \mathbf{l}'', \mathbf{l}'^\top T_3 \mathbf{l}''] \mathbf{u} = 0. \quad (11.59)$$

The nine point-point-point matching constraints among image points $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{u}''$ in, respectively, the first, second and third view can be obtained by substituting any row of matrix $S(\mathbf{u}')$ for $\mathbf{l}'$ and any row of $S(\mathbf{u}'')$ for $\mathbf{l}''$.

The trifocal tensor $\{T_1, T_2, T_3\}$ has $3^3 = 27$ parameters but is defined only up to an overall scale, yielding 26 parameters. However, these parameters satisfy 8 non-linear relations, yielding only 18 free parameters. While we will not discuss these non-linear relations, note that for two views we only had a single non-linear relation, $\det F = 0$.

Given multiple correspondences in three views, the trifocal tensor can be estimated by solving the (possibly overdetermined) system of equations (11.57) or (11.59), which is linear in the components of the tensor. Here, preconditioning described in Section 11.2.3 is essential.

If the trifocal tensor is known then the projection matrices corresponding to individual cameras can be computed from the tensor. The trifocal tensor expresses relation between images and is independent of the particular 3D projection transform. This implies that the projection matrices corresponding to cameras can be computed up to a projective ambiguity. The algorithm for decomposing the trifocal tensor into three projection matrices can be found in [Hartley and Zisserman, 2003].

11.6.1 Stereo correspondence algorithms

We have seen in Section 11.5.1 that much can be learned about the geometry of a 3D scene if it is known which point from one image corresponds to a point in a second image. The solution of this **correspondence problem** is a key step in any photogrammetric, stereo vision, or motion analysis task. Here we describe how the same point can be found in two images if the same scene is observed from two different viewpoints. Of course, it is assumed that two images overlap and thus the corresponding points are sought in this overlapping area.

Some methods are based on the assumption that images constitute a linear (vector) space (e.g., eigenimages or linear interpolation in images [Werner et al., 1995; Ullman and Basri, 1991]); this linearity assumption is not valid for images in general [Beymer and Poggio, 1996], but some authors have overlooked this fact. The structure of a vector space assumes that the i^{th} component of one vector must refer to the i^{th} component of another; this assumes that the correspondence problem has been solved.

Automatic solution of the correspondence problem is an evergreen computer vision topic, and the pessimistic conclusion is that it is not soluble in the general case at all. The trouble is that the problem is inherently ambiguous. Imagine an extreme case, e.g., a scene containing a white, nontextured, flat object; its image constitutes a large region with uniform brightness. When corresponding points are sought in left and right images of the flat object there are not any features that could distinguish them. Another unavoidable difficulty in searching for corresponding points is the **self-occlusion** problem, which occurs in images of non-convex objects. Some points that are visible by the left camera are not visible by the right camera and vice versa (Figure 11.14).

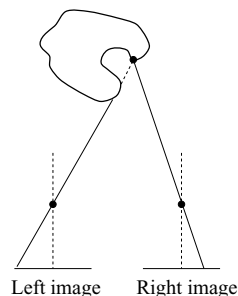


Figure 11.14: Self-occlusion makes search for some corresponding points impossible.
© Cengage Learning 2015.

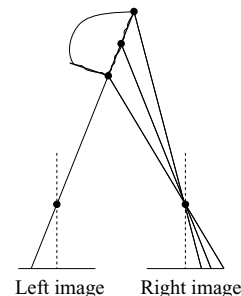


Figure 11.15: Exception from the uniqueness constraint. © Cengage Learning 2015.

Fortunately, uniform intensity and self-occlusion are rare, or at least uncommon, in scenes of practical interest. Establishing correspondence between projections of the same point in different views is based on finding image characteristics that are similar in both views, and the local similarity is calculated.

The inherent ambiguity of the correspondence problem can in practical cases be reduced using several constraints. Some of these follow from the geometry of the image capturing process, some from photometric properties of a scene, and some from prevailing object properties in our natural world. A vast number of different stereo correspondence algorithms have been proposed. We will give here only a concise taxonomy of approaches to finding correspondence—not all the constraints are used in all of them [Klette et al., 1996].

The first group of constraints depends mainly on the geometry and the photometry of the image capturing process.

Epipolar constraint: This says that the corresponding point can only lie on the epipolar line in the second image. This reduces the potential 2D search space into 1D. The epipolar constraint was explained in detail in Section 11.5.

Uniqueness constraint: This states that, in most cases, a pixel from the first image can correspond to at most one pixel in the second image. The exception arises when two or more points lie on one ray coming from the first camera and can be seen as separate points from the second. This case, which arises in the same way as self-occlusion, is illustrated in Figure 11.15.

Symmetry constraint: If the left and right images are interchanged then the same set of matched pairs of points has to be obtained.

Photometric compatibility constraint: This states that intensities of a point in the first and second images are likely to differ only a little. They are unlikely to be exactly the same due to the mutual angle between the light source, surface normal, and viewer differing, but the difference will typically be small and the views will not differ much. Practically, this constraint is very natural to image-capturing conditions. The advantage is that intensities in the left image can be transformed into intensities in the right image using very simple transformations.

Geometric similarity constraints: These build on the observation that geometric characteristics of the features found in the first and second images do not differ much (e.g., length or orientation of the line segment, region, or contour).

The second group of constraints exploits some common properties of objects in typical scenes.

Disparity smoothness constraint: This claims that disparity changes slowly almost everywhere in the image. Assume two scene points $\mathbf{p}$ and $\mathbf{q}$ are close to each other, and denote the projection of $\mathbf{p}$ into the left image as $\mathbf{p}_L$ and into the right image as $\mathbf{p}_R$, and $\mathbf{q}$ similarly. If we assume that the correspondence between $\mathbf{p}_L$ and $\mathbf{p}_R$ has been established, then the quantity

$$\left| |\mathbf{p}_L - \mathbf{p}_R| - |\mathbf{q}_L - \mathbf{q}_R| \right|$$

(the absolute disparity difference) should be small.

Feature compatibility constraint: This places a restriction on the origin of matched points. Points can match only if they have the same physical origin—for example, object surface discontinuity, border of a shadow cast by some objects, occluding boundary or specular boundary. Notice that edges in an image caused by specular or self-occlusion cannot be used to solve the correspondence problem, as they move with changing viewpoint. On the other hand, self-occlusion caused by abrupt discontinuity of the surface can be identified—see Figure 11.16.

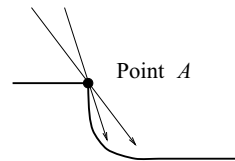


Figure 11.16: Self-occlusion due to abrupt surface discontinuity can be detected. © Cengage Learning 2015.

Disparity search range: This constrains the lengths of the search in artificial methods that seek correspondence.

Disparity gradient limit: This constraint originates from psycho-physical experiments in which it is demonstrated that the human vision system can only fuse stereo images if the disparity change per pixel is smaller than some limit. The constraint is a weak version of the disparity smoothness constraint.

Ordering constraint: This says that for surfaces of similar depth, corresponding feature points typically lie in the same order on the epipolar line (Figure 11.17a). If there is a narrow object much closer to the camera than its background, the order can be changed (Figure 11.17b). It is easy to demonstrate violation of this ordering constraint: hold two forefingers vertically, almost aligned but at different depths in front of your eyes. Closing the left eye and then the right eyes interchanges the left/right order of the fingers. The ordering constraint is violated only rarely in practice.

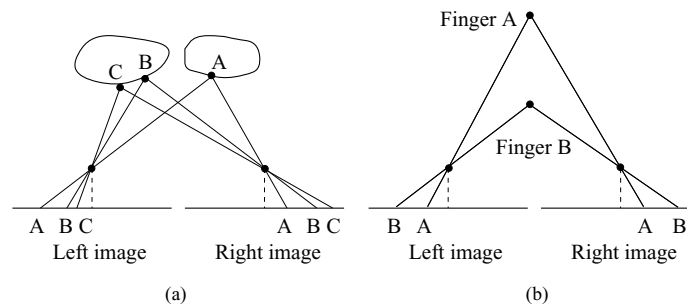


Figure 11.17: Epipolar lines. (a) Corresponding points lie in the same order on epipolar lines. (b) This rule does not hold if there is a big discontinuity in depths. © Cengage Learning 2015.

All these constraints have been of use in one or more existing stereo correspondence algorithms; we present here a taxonomy of such algorithms. From the historical point of view, correspondence algorithms for stereopsis were and still are driven by two main paradigms:

1. Low-level, correlation-based, bottom-up methods.
2. High-level, feature-based, top-down methods.

Initially, it was believed that higher-level features such as corners and straight line segments should be automatically identified, and then matched. This was a natural development from photogrammetry, which has been using feature points identified by human operators since the beginning of the twentieth century.

Psychological experiments with **random dot stereograms** performed by Julesz [Julesz, 1990] generated a new view: these experiments show that humans do not need to create monocular features before binocular depth perception can take place. A random dot stereogram is created in the following way: a left image is entirely random, and the right image is created from it in a consistent way such that some part of it is shifted according to disparity of the desired stereo effect. The viewer must glare at the random dot stereogram from a distance of about 20 centimeters. Such ‘random dot stereograms’ have been widely published under the name ‘3D images’ in many popular magazines.

Correlation-based block matching

Correlation-based correspondence algorithms use the assumption that pixels in correspondence have very similar intensities. The intensity of an individual pixel does not give sufficient information, as there are typically many potential candidates with similar intensity and thus intensities of several neighboring pixels are considered. Typically, a 5×5 or 7×7 or 3×9 window may be used. These methods are sometimes called **area-based stereo**. Larger search windows yield higher discriminability.

We shall illustrate the approach with a simple algorithm called **block matching** [Klette et al., 1996]. Assuming the canonical stereo setup with parallel optical axes of both cameras, the basic idea of the algorithm is that all pixels in the window (called a block) have the same disparity, meaning that one and only one disparity is computed for each block. One of the images, say the left, is tiled into blocks, and a search for correspondence in the right image is conducted for each of these in the right image. The measure of similarity between blocks can be, e.g., the mean square error of the intensity, and the disparity is accepted for the position where the mean square error is minimal. Maximal change of position is limited by the disparity limit constraint. The mean square error can have more than one minimum, and in this case an additional constraint is used to cope with ambiguity.

The result does not obey the symmetry constraint, ordering constraint and gradient limit constraint because the result is not a one-to-one matching.

Another relevant approach is that of Nishihara [Nishihara, 1984], who observes that an algorithm attempting to correlate individual pixels (by, e.g., matching zero-crossings [Marr and Poggio, 1979]) is inclined towards poor performance when noise causes the detected location of such features to be unreliable. Nishihara notes that the sign (and magnitude) of an edge detector response is likely to be a much more stable property to match than the edge or feature locations, and devises an algorithm that simultaneously exploits a scale-space matching attack.

The approach is to match large patches at a large scale, and then refine the quality of the match by reducing the scale, using the coarser information to initialize the finer-grained match. An edge response is generated at each pixel of both images at a large scale (see Section 5.3.4), and then a large area of the left (represented by, say, its central pixel)

is correlated with a large area of the right. This can be done quickly and efficiently by using the fact that the correlation function peaks very sharply at the correct position of a match, and so a small number of tests permits an ascent to a maximum of a correlation measure. This coarse area match may then be iteratively refined to any desired resolution, using the knowledge from the coarser scale as a clue to the correct disparity at a given position. At any stage of the algorithm, therefore, the surfaces in view are modeled as square prisms of varying height; the area of the squares may be reduced by performing the algorithm at a finer scale—for tasks such as obstacle avoidance it is possible that only coarse scale information is necessary, and there will be a consequent gain in efficiency.

Any stereo matching algorithm can be boosted by casting random-dot light patterns on the scene to provide patterns to match even in areas of the scene that are texturally uniform. The resulting system has been demonstrated in use in robot guidance and bin-picking applications, and has been implemented robustly in real time.

Feature-based stereo correspondence

Feature-based correspondence methods use interest points. Characteristically, these are pixels on edges, lines, corners, etc., and correspondence is sought according to properties of such features as, e.g., orientation along edges, or lengths of line segments. The advantages of feature-based methods over intensity-based correlation are:

- Feature-based methods are less ambiguous since the number of potential candidates for correspondence is smaller.
- The resulting correspondence is less dependent on photometric variations in images.
- Disparities can be computed with higher precision; features can be sought in the image to subpixel precision.

We present one early example of a feature-based correspondence method—the **PMF algorithm**, named after its inventors [Pollard et al., 1985]. It proceeds by assuming that a set of feature points has been extracted from each image by some interest operator. The output is a correspondence between pairs of such points. In order to do this, three constraints are applied: the epipolar constraint, the uniqueness constraint, and the disparity gradient limit constraint.

The first two constraints are common to all such algorithms—the third, however, of stipulating a disparity gradient limit, was its novelty. The **disparity gradient** measures the relative disparity of two pairs of matching points.

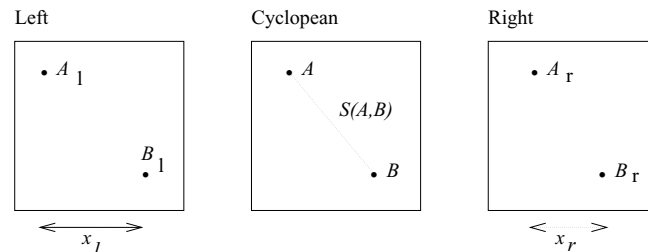


Figure 11.18: Definition of the disparity gradient. © Cengage Learning 2015.

Suppose (Figure 11.18) that a point A (B) in 3D appears as $A_l = (a_{xl}, a_y)$ ($B_l = (b_{xl}, b_y)$) in the left image and $A_r = (a_{xr}, a_y)$ ($B_r = (b_{xr}, b_y)$) in the right (the epipolar constraint requires the y coordinates to be equal); the **cyclopean** image is defined as that given by their average coordinates

$$A_c = \left(\frac{a_{xl} + a_{xr}}{2}, a_y \right), \quad (11.60)$$

$$B_c = \left(\frac{b_{xl} + b_{xr}}{2}, b_y \right) \quad (11.61)$$

and their **cyclopean separation** S is given by their distance apart in this image

$$\begin{aligned} S(A, B) &= \sqrt{\left[\left(\frac{a_{xl} + a_{xr}}{2} \right) - \left(\frac{b_{xl} + b_{xr}}{2} \right) \right]^2 + (a_y - b_y)^2} \\ &= \sqrt{\frac{1}{4} [(a_{xl} - b_{xl}) + (a_{xr} - b_{xr})]^2 + (a_y - b_y)^2} \\ &= \sqrt{\frac{1}{4} (x_l + x_r)^2 + (a_y - b_y)^2}. \end{aligned} \quad (11.62)$$

The difference in disparity between the matches of A and B is

$$\begin{aligned} D(A, B) &= (a_{xl} - a_{xr}) - (b_{xl} - b_{xr}) \\ &= (a_{xl} - b_{xl}) - (a_{xr} - b_{xr}) \\ &= x_l - x_r. \end{aligned} \quad (11.63)$$

The disparity gradient of the pair of matches is then given by the ratio of the disparity difference to the cyclopean separation:

$$\begin{aligned} \Gamma(A, B) &= \frac{D(A, B)}{S(A, B)} \\ &= \frac{x_l - x_r}{\sqrt{\frac{1}{4} (x_l + x_r)^2 + (a_y - b_y)^2}}. \end{aligned} \quad (11.64)$$

Given these definitions, the constraint exploited is that, in practice, the disparity gradient Γ can be expected to be limited; in fact, it is unlikely to exceed 1. This means that very small differences in disparity are not acceptable if the corresponding points are extremely close to each other in 3D—this seems an intuitively reasonable observation, and it is supported by physical evidence [Pollard et al., 1985]. A solution to the correspondence problem is then extracted by a relaxation process in which all possible matches are scored according to whether they are supported by other (possible) matches that do not violate the stipulated disparity gradient limit. High-scoring matches are regarded as correct, permitting firmer evidence to be extracted about subsequent matches.

Algorithm 11.2: PMF stereo correspondence

1. Extract features to match in left and right images. These may be, for example, edge pixels.

2. For each feature in the left (say) image, consider its possible matches in the right; these are defined by the appropriate epipolar line.
3. For each such match, increment its likelihood score according to the number of other possible matches found that do not violate the chosen disparity gradient limit.
4. Any match which is highest scoring for both the pixels composing it is now regarded as correct. Using the uniqueness constraint, these pixels are removed from all other considerations.
5. Return to step 2 and re-compute the scores taking account of the definite match derived.
6. Terminate when all possible matches have been extracted.

Note here that the epipolar constraint is used in step 2 to limit to one dimension the possible matches of a pixel, and the uniqueness constraint is used in step 4 to ensure that a particular pixel is never used more than once in the calculation of a gradient.

The scoring mechanism has to take account of the fact that the more remote two (possible) matches are, the more likely they are to satisfy the disparity gradient limit. This is catered for by:

- Considering only matches that are ‘close’ to the one being scored. In practice it is typically adequate to consider only those inside a circle of radius equal to 7 pixels, centered at the matching pixels (although this number depends on the precise geometry and scene in hand).
- Weighting the score by the reciprocal of its distance from the match being scored. Thus more remote pairs, which are more likely to satisfy the limit by chance, count for less.

This algorithm has been demonstrated to work successfully. It is also attractive because it lends itself to parallel implementation and could be extremely fast on suitable hardware. It has a drawback (along with a number of similar algorithms) in that horizontal line segments are hard to match; they often move across adjacent rasters and, with parallel camera geometry, any point on one such line can match any point on the corresponding line in the other image.

Since PMF was developed many other algorithms of varying complexity have been proposed. Two computationally efficient and simple to implement algorithms utilize either optimization techniques called dynamic programming [Gimel’farb, 1999] or confidently stable matching [Sara, 2002]. An extensive list of stereo matching algorithm is maintained at <http://cat.middlebury.edu/stereo/>.

11.6.2 Active acquisition of range images

It is extremely difficult to extract 3D shape information from intensity images of real scenes directly. Another approach—‘shape from shading’—will be explained in Section 11.7.1.

One way to circumvent these problems is to measure distances from the viewer to points on surfaces in the 3D scene explicitly; such measurements are called **geometric signals**, i.e., a collection of 3D points in a known coordinate system. If the surface relief is measured from a single viewpoint, it is called a **range image** or a **depth map**. Such explicit 3D information, being closer to the geometric model that is sought, makes geometry recovery easier.¹

Two steps are needed to obtain geometric information from a range image:

1. The range image must be captured; this procedure is discussed in this section.
2. Geometric information must be extracted. Features are sought and compared to a selected 3D model. The selection of features and geometric models leads to one of the most fundamental problems in computer vision: how to represent a solid shape [Koenderink, 1990].

The term **active sensor** refers to a sensor that uses and controls its own images—the term ‘active’ means that the sensor uses and controls electromagnetic energy, or more specifically illumination, for measuring a distance between scene surfaces and the ‘observer’. An active sensor should not be confused with the active perception strategy, where the sensing subject plans how to look at objects from different views.

RADAR (RAdio Detecting And Ranging) and **LIDAR** (LIght Detecting And Ranging) in one measurement yield the distance between the sensor and a particular point in a scene. The sensor is mounted on an assembly that allows movement around two angles, azimuth Θ and tilt Φ , corresponding to spherical coordinates. The distance is proportional to the time interval between the emission of energy and the echo reflected from the measured scene object—the elapsed time intervals are very short, so very high precision is required. For this reason, the phase difference between emitted and received signals is often used.

RADAR emits electromagnetic waves in meter, centimeter, or millimeter wavelength bands. Aside from military use, it is frequently used for navigation of autonomous guided vehicles.

LIDAR often uses laser as a source of a focused light beam. The higher the power of the laser, the stronger is the reflected signal and the more precise the measured range. If LIDAR is required to work in an environment together with humans, then the energy has an upper limit, due to potential harm to the unprotected eye. Another factor that influences LIDAR safety is the diameter of the laser beam: if it is to be safe, it should not be focused too much. LIDARs have trouble when the object surface is almost tangential to the beam, as very little energy reflects back to the sensor in this case. Measurements of specular surfaces are not very accurate, as they scatter the reflected light; while transparent objects (obviously) cannot be measured with optical lasers. The advantage of LIDAR is a wide range of measured distances, from a tenth of a millimeter to several kilometers; the accuracy of the measured range is typically around 0.01 millimeter. LIDAR provides one range in an instant. If a whole range image is to be captured, the measurement takes several tenths of a seconds as the whole scene is scanned.

Another principle of active range imaging is **structured light triangulation**, where we employ a geometric arrangement similar to that used for stereo vision, with optical

¹There are techniques that measure full 3D information directly, such as mechanical coordinate-measuring machines (considered in Chapter 12) or computer tomography.

axes. One camera is replaced by an illuminant that yields a light plane perpendicular to the epipolars; the image-capturing camera is at a fixed distance from the illuminant. Since there is only one significantly bright point on each image line, the correspondence problem that makes passive stereo so problematic is avoided, although there will still be problems with self-occlusion in the scene. Distance from the observer can easily be calculated as in Figure 11.11. To capture a whole-range image, the rod with camera and illuminant should be made to move mechanically relative to the scene, and the trace of the laser should gradually illuminate all points to be measured. The conduct of the movement, together with the processing of several hundred images (i.e., one image for each distinct position of the laser-stripe) takes some time, typically from a couple of seconds to about a minute. Faster laser stripe range finders find a bright point corresponding to the intersection of a current image line using special-purpose electronics.

Figure 11.19 shows a view of such a scanner together with a target object (a wooden toy—a rabbit). The image seen by the camera with the distinct bright laser stripe is in Figure 11.20a, and the resulting range image is shown in Figure 11.20b.

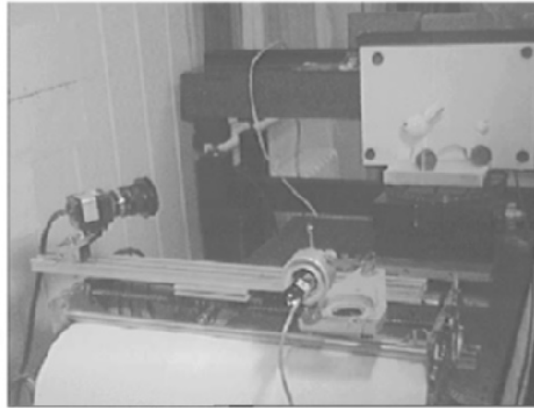


Figure 11.19: Laser plane range finder. The camera is on the left side, the laser diode on the bottom. *Courtesy of T. Pajdla, Czech Technical University, Prague.*

In some applications, a range image is required in an instant, typically meaning one TV frame; this is especially useful for capturing range images of moving objects, e.g., moving humans (see Section 10.8). One possibility is to illuminate the scene by several stripes at once and code them; Figure 11.21a shows a human hand lit by a binary cyclic code pattern such that the local configuration of squares in the image allows to us to decode which stripe it is. In this case, the pattern with coded stripes is projected from a 36×24 mm slide using a standard slide projector. The resulting range image does not provide as many samples as in the case of a moving laser stripe—in our case only 64×80 , see Figure 11.21b.

It is possible to acquire a dense range sample as in the laser stripe case in one TV frame; individual stripes can be encoded using spectral colors and the image captured by a color TV camera [Smutný, 1993].

There are other technologies available. One is sonar, which uses ultrasonic waves as an energy source. Sonars are used in robot navigation for close-range measurements,

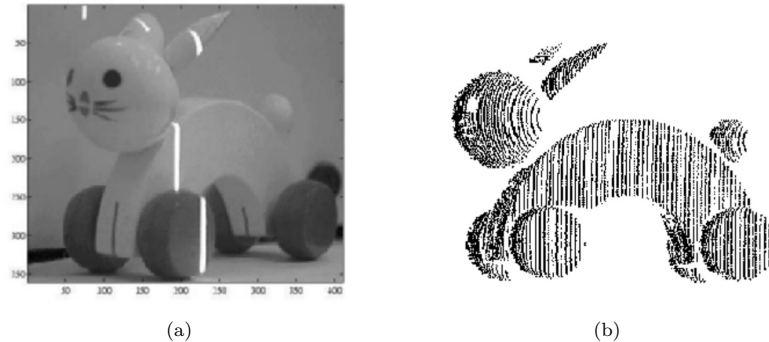


Figure 11.20: Measurement using a laser-stripe range finder. (a) The image seen by a camera with a bright laser stripe. (b) Reconstructed range image displayed as a point cloud. *Courtesy of T. Pajdla, Czech Technical University, Prague.*

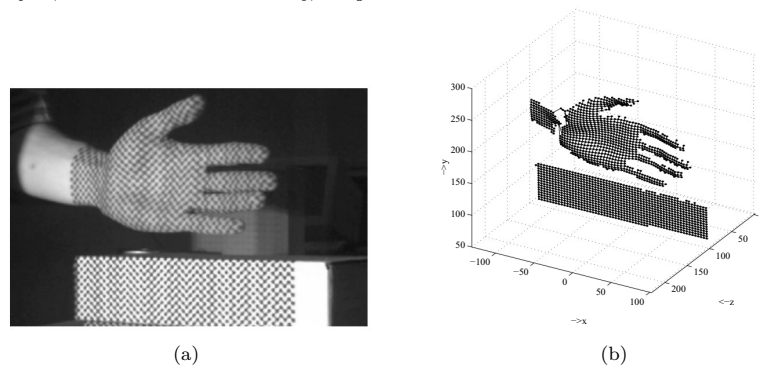


Figure 11.21: Binary-coded range finder. (a) The captured image of a hand. (b) Reconstructed surface. *Courtesy of T. Pajdla, Czech Technical University, Prague.*

but a disadvantage is that measurements are typically very noisy. The second principle is Moiré interferometry [Klette et al., 1996], in which two periodic patterns, typically stripes, are projected on the scene. Due to interference, the object is covered by a system of closed, non-intersecting curves, each of which lies in a plane of constant distance from the viewer. Distance measurements obtained are only relative, and absolute distances are unavailable. The properties of Moiré curves are very similar to height contours on maps.

11.7 3D information from radiometric measurements

We have argued several times in this book that the image formation from the radiometric point of view is well understood (Section 3.4.4). We have also explained that the inverse task in which the input is the intensity image and the output is 3D properties of a surface in the scene is ill-posed and extremely difficult to solve in most cases. Instead of solving it, a side-step is taken and objects in the image are segmented using some

semantic information, but not directly the image formation physics. Nevertheless, there are special situations in which the inverse task to image formation has a solution. The first approach is **shape from shading**, and the second one **photometric stereo**.

11.7.1 Shape from shading

The human brain makes use of clues from shadows and shading in general. Not only do detected shadows give a clear indication of where occluding edges are, and the possible orientation of their neighboring surfaces, but general shading properties are of great value in deducing depth. A fine example of this is a photograph of a face; from a straight-on, 2D representation, our brains make good guesses about the probable lighting model, and then deductions about the 3D nature of the face—for example, deep eye sockets and protuberant noses or lips are often recognizable without difficulty.

Recall that the intensity of a particular pixel depends on the light source(s), surface reflectance properties, and local surface orientation expressed by a surface normal $\mathbf{n}$. The aim of shape from shading is to extract information about normals of surfaces in view solely on the basis of an intensity image. If simplifying assumptions are made about illumination, surface reflectance properties, and surface smoothness, the shape from shading task has proven to be solvable. The first computer-vision related formulation comes from Horn [Horn, 1970, 1975]. Techniques similar to shape from shading were earlier proposed independently in photoclinometry [Rindfleisch, 1966], when astrogeologists wanted to measure steepness of slopes on planets in the solar system from intensity images observed by terrestrial telescopes.

Incremental propagation from surface points of known height

The oldest, and easiest to explain, method develops a solution along a space curve. This is also called the characteristic strip method.

We can begin to analyze the problem of global shape extraction from shading information when the reflectance function and the lighting model are both known perfectly [Horn, 1990]. Even given these constraints, it should be clear that the mapping ‘surface orientation to brightness’ is many-to-one, since many orientations can produce the same point intensity. Acknowledging this, a particular brightness can be produced by an infinite number of orientations that can be plotted as a (continuous) closed curve in gradient space. An example for the simple case of a light source directly adjacent to the viewer, incident on a matte surface, is shown in Figure 11.22—two points lying on the same curve (circles in this case) indicate two different orientations that will reflect light of the same intensity, thereby producing the same pixel gray-level.

The original formulation [Horn, 1970] to the general shape from shading task assumes a Lambertian surface, one distant point light source, a distant observer, and no inter-reflections in the scene. The proposed method is based on the notion of a **characteristic strip**: suppose that we have already calculated coordinates of a surface point $[x, y, z]^T$ and we want to propagate the solution along an infinitesimal step on the surface, e.g., taking small steps δx and δy , then calculating the change in height δz . This can be done if the components of the surface gradient p, q are known. For compactness we use an index notation, and express $p = \delta z / \delta x$ as z_x , and $\delta^2 x / \delta x^2$ as z_{xx} . The infinitesimal change of height is

$$\delta z = p \delta x + q \delta y. \quad (11.65)$$

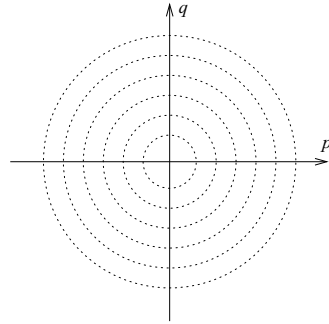


Figure 11.22: Reflectance map for a matte surface—the light source is adjacent to the viewer. © Cengage Learning 2015.

The surface is followed stepwise, with values of p, q being traced along with x, y, z . Changes in p, q are calculated using second derivatives of height $r = z_{xx}$, $s = z_{xy} = z_{yx}$, $t = z_{yy}$

$$\delta p = r \delta x + s \delta y \quad \text{and} \quad \delta q = s \delta x + t \delta y. \quad (11.66)$$

Consider now the image irradiance equation $E(x, y) = R(p, q)$, equation (3.96), and differentiate with respect to x, y to obtain the brightness gradient

$$E_x = r R_p + s R_q \quad \text{and} \quad E_y = s R_p + t R_q. \quad (11.67)$$

The direction of the step $\delta x, \delta y$ can be chosen arbitrarily

$$\delta x = R_p \xi \quad \text{and} \quad \delta y = R_q \xi. \quad (11.68)$$

The parameter ξ changes along particular solution curves. Moreover, the orientation of the surface along this curve is known; thus it is called a characteristic strip.

We can now express changes of gradient $\delta p, \delta q$ as dependent on gradient image intensities, which is the crucial trick. A set of ordinary differential equations can be generated by considering equations (11.66) and (11.67); dot denotes differentiation with respect to ξ

$$\dot{x} = R_p, \quad \dot{y} = R_q, \quad \dot{z} = p R_p + q R_q, \quad \dot{p} = E_x, \quad \dot{q} = E_y. \quad (11.69)$$

There are points on the surface for which the surface orientation is known in advance, and these provide boundary conditions during normal vector calculations. These are

- Points of a surface **occluding boundary**; an occluding boundary is a curve on the surface due to the surface rolling away from the viewer, i.e., the set of points for which the local tangent plane coincides with the direction towards the viewer. The surface normal at such a boundary can be uniquely determined (up to a sign), as it is parallel to the image plane and perpendicular to the direction towards the viewer. This normal information can be propagated into the recovered surface patch from the occluding boundary. Although the occlusion boundary uniquely constrains surface orientation, it does not constrain the solution sufficiently to recover depth uniquely [Oliensis, 1991].
- Singular points in the image; we have seen that at most surface points gradient is not fully constrained by image intensities. Suppose that the reflectance function

$R(p, q)$ has a global maximum, so $R(p, q) < R(p_0, q_0)$ for all $[p, q] \neq p_0, q_0$. This maximum corresponds to singular points in the image

$$E(x_0, y_0) = R(p_0, q_0). \quad (11.70)$$

Here, the surface normal is parallel to the direction towards the light source. Singular points are in general sources and sinks of characteristic stripes.

Direct implementation of this method does not yield particularly good results, due to numerical instability of independent integrations along the stripes [Horn, 1990].

Global optimization methods

These methods are formulated as a variational task in which the whole image plays a role in the chosen functional. Results obtained are in general better than those generated by incremental methods.

We already know that under the simplifying conditions for recovery of surface normals from intensities (stated in Section 3.4.5), the image irradiance equation (3.96) relates image irradiance E and surface reflection R as

$$E(x, y) = R(p(x, y), q(x, y)). \quad (11.71)$$

The task is to find the surface height $z(x, y)$ given the image $E(x, y)$ and reflectance map $R(p, q)$.

Now, presented with an intensity image, a locus of possible orientations for each pixel can be located in gradient space, immediately reducing the number of possible interpretations of the scene. Of course, at this stage, a pixel may be part of any surface lying on the gradient space contour; to determine which, another constraint needs to be deployed. The key to deciding which point on the contour is correct is to note that ‘almost everywhere’ 3D surfaces are smooth, in the sense that neighboring pixels are very likely to represent orientations whose gradient space positions are also very close. This additional constraint allows a relaxation process to determine a best-fit (minimum-cost) solution to the problem. The details of the procedure are very similar to those used to extract optical flow, and are discussed more fully in Section 16.2.1, but may be summarized as follows.

Algorithm 11.3: Reconstructing shape from shading

1. For each pixel (x_i, y_i) , select an initial guess to orientation $p^0(x_i, y_i)$, $q^0(x_i, y_i)$. The subscript i addresses image pixels.
2. Apply two constraints:
 - (a) The observed intensity $f(x_i, y_i)$ should be close to that predicted by the reflectance map $R(p(x_i, y_i), q(x_i, y_i))$ derived from foreknowledge of the lighting and surface properties.
 - (b) Functions p and q vary smoothly—therefore their gradients $(\nabla p)^2$ and $(\nabla q)^2$ should be small.

3. Minimize the energy E

$$E = \sum_i \left(f(x_i, y_i) - R(p(x_i, y_i), q(x_i, y_i)) \right)^2 + \lambda \iint_{\Omega} (\nabla p)^2 + (\nabla q)^2 dx dy. \quad (11.72)$$

The first term of equation (11.72) is the intensity ‘knowledge’. The second term is integrated along the whole (continuous) domain Ω . After discretization, the minimization is solved using the conjugate gradient descent method [Szeliski, 1990].

The significant work in this area is due to Horn [Horn, 1975] and Ikeuchi [Ikeuchi and Horn, 1981] and pre-dated the publication of Marr’s theory, being complete in the form presented here by 1980. Shape from shading, as implemented by machine vision systems, is often not as reliable as other ‘shape from’ techniques, since it is so easy to confuse with reflections, or for it to fail through poorly modeled reflectance functions. This observation serves to reinforce the point that the human recognition system is very powerful, since in deploying elaborate knowledge it does not suffer these drawbacks.

Local shading analysis

Local shading analysis methods use just a small neighborhood of the current point on the surface, and seek a direct relation between the differential surface structure and the local structure of the corresponding intensity image. The surface is considered as a set of small neighborhoods, each defined in some local neighborhood of one of its points. Only an estimate of local surface orientation is available, not information about the height of a particular surface point.

The main advantage of local shading analysis is that it provides surface-related information to higher-level vision algorithms from a single monocular-intensity image without any need to reconstruct the surface in explicit depth form [Sara, 1995]. This is possible because the intensity image is closely related to local surface orientation. The surface normal and the shape operator (‘curvature matrix’) form a natural shape model that can be recovered from an intensity image by local computations except it is not usually possible to recover the signs of principal surface curvatures which result in a four-fold convex/concave/elliptic/hyperbolic ambiguity. This approach is, of course, much faster than the solution propagation or global variational methods but it does not provide local information on convexity and ellipticity of the surface.

The fundamental contribution to local shading analysis comes from Pentland [Pentland, 1984]; an overview can be found in [Pentland and Bichsel, 1994]. In addition, Šára [Sara, 1994] demonstrates:

1. Local surface orientation up to a sign and Gaussian curvature sign can be determined uniquely at occlusion boundaries. Further, orientation on a self-shadowing boundary can also be determined uniquely; self-shadowing contours are thus a rich source of unambiguous information about the surface.
2. The differential properties of isophotes (curves of constant image intensity) are closely related to the properties of the underlying surface. Isophotes are projections of curves of constant slant from the light direction if the surface reflectance is space invariant, or the illuminant is located at the vantage point.

11.7.2 Photometric stereo

Photometric stereo is a method that recovers surface orientation unambiguously, assuming a known reflectance function [Woodham, 1980]. Consider a particular Lambertian surface with varying albedo ρ . The key idea of photometric stereo is to look at the surface from one fixed viewing direction while changing the direction of incident illumination. Assume we have three or more such images of the Lambertian surface; then the surface normals can be uniquely determined based on the shading variations in the observed images if we know the illuminants' positions.

The lines of constant reflectance on the surface correspond to lines of constant irradiation E in the image (called also isophotes). The local surface orientation $\mathbf{n} = [-p, -q, 1]$ is constrained along a second-order curve in the reflectance map. For different illumination directions, the observed reflectance map $R(p, q)$ changes. This provides an additional constraint on possible surface orientation that is another second-order polynomial. Two views corresponding to two distinct illumination directions are not enough to determine the surface orientation $[-p, -q, 1]$ uniquely, and a third view is needed to derive a unique solution. If more than three distinct illuminations are at hand, an over-determined set of equations can be solved.

A practical setup for image capture consists of one camera and K point illumination sources, $K \geq 3$, with known intensities and illumination directions $L_1, \dots, L_K$. Only one light source is active at any one time. The setup should be photometrically calibrated to take into account light source intensities, particular camera gain, and offset; such a calibration is described in [Haralick and Shapiro, 1993]. After photometric calibration, the images give K estimates of image irradiance $E_i(x, y)$; $i = 1, \dots, K$.

If not all light is reflected from a surface, then albedo ρ , $0 \leq \rho \leq 1$, occurs in the image irradiance as shown in equation (3.94). For a Lambertian surface the image irradiance equation simplifies to

$$E(x, y) = \rho R(p, q). \quad (11.73)$$

Recall equation (3.93) (called the cosine law), showing that the reflectance map of a Lambertian surface is given by the dot product of the surface normal $\mathbf{n}$ and the direction of the incident light L_i . If the surface reflectance map is substituted into equation (11.73), we get K image irradiance equations

$$E_i(x, y) = \rho L_i^\top \mathbf{n}, \quad i = 1, \dots, K. \quad (11.74)$$

For each point x, y in the image we get a vector of image irradiances $\mathbf{E} = [E_1, \dots, E_K]^\top$. The light directions can be written in the form of a $K \times 3$ matrix

$$L = \begin{bmatrix} L_1^\top \\ \vdots \\ L_K^\top \end{bmatrix}. \quad (11.75)$$

At each image point, the system of image irradiance equations can be written as

$$\mathbf{E} = \rho L \mathbf{n}. \quad (11.76)$$

The matrix L does not depend on the pixel position in the image, and we can thus derive a vector representing simultaneously surface albedo and a local surface orientation.

If we have three light sources, $K = 3$, we can derive a solution by inverting the regular matrix L

$$\rho \mathbf{n} = L^{-1} \mathbf{E} \quad (11.77)$$

and the unit normal is

$$\mathbf{n} = \frac{L^{-1} \mathbf{E}}{\|L^{-1} \mathbf{E}\|}. \quad (11.78)$$

For more than three light sources, the pseudo-inverse of a rectangular matrix is determined to get a solution in the least-square sense

$$\mathbf{n} = \frac{(L^T L)^{-1} L^T \mathbf{E}}{\|(L^T L)^{-1} L^T \mathbf{E}\|}. \quad (11.79)$$

Note that the pseudo-inversion or inversion in equation (11.78) must be repeated for each image pixel x, y to derive an estimate of the corresponding normal.

It has been shown how to position the lights to obtain photometric stereo results with minimal error [Drbohlav and Chantler, 2005a], and how to calibrate lights in the case that the only information available are positions of highlights in images [Drbohlav and Chantler, 2005b].

11.8 Summary

- 3D vision aims at inferring 3D information from 2D scenes, a task with embedded geometric and radiometric difficulties. The geometric problem is that a single image does not provide enough information about 3D structures, and the radiometric problem is the complexity of the physical process of intensity image creation. This process is complex, and typically not all input parameters are known precisely.
- **3D vision tasks**
 - There are several approaches to 3D vision, which may be categorized as **bottom-up** (or reconstruction) or **top-down** (model-based vision).
 - *Marr's theory*, formulated in the late 1970s, is an example of the bottom-up approach. The aim is to reconstruct qualitative and quantitative 3D geometric descriptions from one or more intensity images under very weak assumptions about objects in the scene.
 - Four representations are ordered in bottom-up fashion: (1) input intensity image(s); (2) primal sketch, representing significant edges in the image in viewer-centered coordinates; (3) 2.5D sketch, representing depth from the observer and local orientation of the surface; and (4) 3D representation, representing object geometry in coordinates related to the objects themselves.
 - The 2.5D sketch is derived from the primal sketch by a variety of techniques called shape from X.
 - 3D representations are very hard to obtain; this step has not been solved in the general case.
 - More recent perception paradigms such as active, purposive, and qualitative vision try to provide a computational model explaining the 'understanding' aspects of vision.

- None has yet led to direct practical applications, but many partial techniques (such as shape from X) are widely used in practice.
- **3D vision, its geometry**
 - 3D perspective geometry is the basic mathematical tool for 3D vision as it explains a pinhole camera.
 - Lines parallel in the 3D world do not project as parallel lines in 2D images.
 - The case of the single-perspective camera permits careful study of calibration of intrinsic and extrinsic camera parameters.
 - Two-perspective cameras constitute stereopsis and allow depth measurements in 3D scenes.
 - Epipolar geometry teaches us that the search for corresponding points is inherently one-dimensional. This can be expressed algebraically using the fundamental matrix.
 - This tool has several applications, such as image rectification, ego-motion estimation from calibrated cameras measurements, 3D Euclidean reconstruction from two fully calibrated cameras, 3D similarity reconstruction from two cameras with only intrinsic calibration parameters known, and 3D projective reconstruction from two uncalibrated cameras.
 - There is a trilinear relation among views of three cameras that is expressed algebraically using a trifocal tensor.
 - The application of the trilinear relation is in epipolar transfer; if two images are known, together with the trifocal tensor, the third perspective image can be computed.
 - The correspondence problem is core to 3D vision; various passive and active techniques to solve it exist.
- **Radiometry and 3D vision**
 - Radiometry informs us about the physics of image formation.
 - If position of illuminants, type, surface reflectance and viewer position are known, something can be learned about depth and scene surface orientation from one intensity image.
 - This task is called shape from shading.
 - The task is ambiguous and numerically unstable. Shape from shading can be understood in the simple case of Lambertian surfaces.
 - There is a practical method that uses one camera and three known illuminants; selective illumination provides three intensity images.
 - Photometric stereo allows a measure of orientation of surfaces.

11.9 Exercises

Short-answer questions

- S11.1 Explain the difference between a bottom-up approach (object reconstruction) to 3D vision as opposed to top-down (model-based).

- S11.2 Explain the basic idea of active vision, and give some examples of how this approach eases vision tasks.
- S11.3 Give examples of perspective images from everyday life. Where do parallel lines in the world not correspond to parallel ones in images?
- S11.4 What are the intrinsic and extrinsic calibration parameters of a single-perspective camera? How are they estimated from known scenes?
- S11.5 Do zoom lenses typically have worse geometric distortion compared to fixed-focal-length lenses? Is the difference significant?
- S11.6 What is the main contribution of epipolar geometry in stereopsis?
- S11.7 Where are epipoles in the case of two cameras with parallel optical axes (the *canonical* configuration for stereopsis)?
- S11.8 What is the difference between the fundamental and essential matrices in stereopsis?
- S11.9 How are mismatches in correspondences treated in stereopsis?
- S11.10 What are the applications of epipolar geometry in computer vision?
- S11.11 Explain the principle, advantages, and applications of a trilinear relation among three cameras. What is epipolar transfer?
- S11.12 Why cannot we get depth if the camera does not translate?
- S11.13 Why cannot we get a correct panorama if the camera does translate?
- S11.14 Stereo correspondence algorithms are typically lost if the left and right images have large regions of uniform brightness. How can depth acquisition still be made possible?
- S11.15 Active range finders (e.g., with a laser plane) suffer from occlusions; some points are not visible by the camera and some are not lit. What are the ways of dealing with this problem?
- S11.16 What is Moiré interferometry? Does it give absolute depth?
- S11.17 Why is the relation between pixel intensity on one side and surface orientation, surface reflectance, illuminant types and position, and viewer position on the other side difficult?
- S11.18 Under which circumstances can the surface orientation be derived from intensity changes in an image?

Problems

- P11.1 This problem relates to Marr's theory, in particular the representation scheme called primal sketch (see Section 11.1.1).
Capture an intensity image (e.g., of an office scene), and run an advanced edge detector (such as Canny's or similar) on it. Threshold the magnitude of the image gradient.
Answer the following questions in an essay: Are the lines which you get the primal sketch? Would it be possible to derive a 2.5D sketch directly from it? How? Discuss what more would you need in the primal sketch. What about multiple scales?
- P11.2 Explain the notion of homogeneous co-ordinates. Is projective transformation linear if expressed in homogeneous co-ordinates? Why are homogeneous co-ordinates often used in robotics to express the kinematics of a manipulating arm? (Hint: Express rotation and translation of an object in 3D space using homogeneous co-ordinates.)
- P11.3 Take a camera with an off-the-shelf lens. Design and perform an experiment to find the intrinsic calibration parameters of it. Design and use an appropriate calibration object (for example, a grid like structure printed by laser printer on paper; you might capture it at different heights by placing it on a box of known height). Discuss the precision of your results. Is the pinhole model of your camera appropriate? (Hint: Look at distortions of a grid as in Figure 3.34.)

- P11.4 Consider the case of two cameras (stereopsis) with baseline $2h$. A scene point lies on the optical axis at depth d from the baseline. Assume that the precision of pixel position measurement x in the image plane is given by dispersion σ^2 . Derive a formula showing the dependence of the precision of depth measurement against dispersion. (Hint: Differentiate d according to x .)
- P11.5 Conduct an experiment with stereo correspondences. For simplicity, capture a pair of stereo images using cameras in canonical configuration (epipolar lines correspond to lines in images), and cut corresponding lines from both images. Visualize the brightness profiles in those lines (for example, using Matlab). First, try to find correspondences in brightness manually. Second, decide if correlation-based or feature-based stereo techniques are more suitable for your case. Program it and test on your profiles.
- P11.6 Conduct a laboratory experiment with photometric stereo. You will need one camera and three light sources. Take some opaque object and measure its surface orientation using photometric stereo (see Section 11.7.2).
- P11.7 Make yourself familiar with solved problems and **Matlab** implementations of selected algorithms provided in the corresponding chapter of the **Matlab Companion** to this text [Svoboda et al., 2008]. The **Matlab Companion** homepage <http://visionbook.felk.cvut.cz> offers images used in the problems, and well-commented **Matlab** code is provided for educational purposes.
- P11.8 Use the **Matlab Companion** [Svoboda et al., 2008] to develop solutions to additional exercises and practical problems provided there. Implement your solutions using **Matlab** or other suitable programming languages.

11.10 References

- Aloimonos Y., editor. *Active Perception*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.
- Aloimonos Y. What I have learned. *CVGIP: Image Understanding*, 60(1):74–85, 1994.
- Aloimonos Y. and Rosenfeld A. Principles of computer vision. In Young T. Y., editor, *Handbook of Pattern Recognition and Image Processing: Computer Vision*, pages 1–15, San Diego, 1994. Academic Press.
- Aloimonos Y. and Shulman D. *Integration of Visual Modules—An Extension of the Marr Paradigm*. Academic Press, New York, 1989.
- Aloimonos Y., editor. Special issue on purposive and qualitative active vision. *CVGIP B: Image Understanding*, 56, 1992.
- Bajcsy R. Active perception. *Proceedings of the IEEE*, 76(8):996–1005, 1988.
- Bertero M., Poggio T., and Torre V. Ill-posed problems in early vision. *IEEE Proceedings*, 76: 869–889, 1988.
- Besl P. J. and Jain R. *Surfaces in range image understanding*. Springer Verlag, New York, 1988.
- Besl P. J. and Jain R. C. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, 1985.
- Beymer D. and Poggio T. Image representations for visual learning. *Science*, 272:1905–1909, 1996.
- Biederman I. Recognition by components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.
- Bowyer K. W. Special issue on directions in CAD-based vision. *CVGIP – Image Understanding*, 55:107–218, 1992.

- Brooks R. A., Greiner R., and Binford T. O. The ACRONYM model-based vision system. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-6*, Tokyo, Japan, pages 105–113, 1979.
- Buxton H. and Howarth R. J. Spatial and temporal reasoning in the generation of dynamic scene representations. In Rodriguez R. V., editor, *Proceedings of Spatial and Temporal Reasoning*, pages 107–115, Montreal, Canada, 1995. IJCAI-95.
- Cornelius H., Šára R., Martinec D., Pajdla T., Chum O., and Matas J. Towards complete free-form reconstruction of complex 3D scenes from an unordered set of uncalibrated images. In Comaniciu D., Mester R., and Kanatani K., editors, *Proceedings of the ECCV Workshop Statistical Methods in Video Processing*, pages 1–12, Heidelberg, Germany, 2004. Springer-Verlag.
- Drbohlav O. and Chantler M. On optimal light configurations in photometric stereo. In *Proceedings of the 10th IEEE International Conference on Computer Vision*, volume II, pages 1707–1712, Beijing, China, 2005a. IEEE Computer Society.
- Drbohlav O. and Chantler M. Can two specular pixels calibrate photometric stereo? In *Proceedings of the 10th IEEE International Conference on Computer Vision*, volume II, pages 1850–1857, Beijing, China, 2005b. IEEE Computer Society.
- Farshid A. and Aggarwal J. K. Model-based object recognition in dense range images—a review. *ACM Computing Surveys*, 25(1):5–43, 1993.
- Faugeras O. D. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA, 1993.
- Faugeras O. D., Luong Q. T., and Maybank S. J. Camera self-calibration: Theory and experiments. In *2nd European Conference on Computer Vision*, Santa Margherita Ligure, Italy, pages 321–333, Heidelberg, Germany, 1992. Springer Verlag.
- Fernyhough J. F. *Generation of qualitative spatio-temporal representations from visual input*. Ph.D. thesis, School of Computer Studies, University of Leeds, Leeds, UK, 1997.
- Fitzgibbon A. W. and Zisserman A. Automatic camera recovery for closed or open image sequences. In *Proceedings of the European Conference on Computer Vision*, volume LNCS 1406, pages 311–326, Heidelberg, Germany, 1998. Springer Verlag.
- Flynn P. J. and Jain A. K. CAD-based computer vision: From CAD models to relational graphs. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 13(2):114–132, 1991.
- Flynn P. J. and Jain A. K. 3D object recognition using invariant feature indexing of interpretation tables. *CVGIP – Image Understanding*, 55(2):119–129, 1992.
- Forsyth D. and Ponce J. *Computer Vision: A Modern Approach*. Prentice Hall, New York, NY, 2003.
- Gimelfarb G. *Handbook of Computer Vision and Applications*, volume 2, Signal processing and pattern recognition, chapter Stereo terrain reconstruction by dynamic programming, pages 505–530. Academic Press, San Diego, California, USA, 1999.
- Gluckman J. and Nayar S. K. Rectifying transformations that minimize resampling effects. In Kasturi R. and Medioni G., editors, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 111–117. IEEE Computer Society, 2001.
- Goad C. Fast 3D model-based vision. In Pentland A. P., editor, *From Pixels to Predicates*, pages 371–374. Ablex, Norwood, NJ, 1986.
- Golub G. H. and Loan C. F. V. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 2nd edition, 1989.

- Haralick R. M. and Shapiro L. G. *Computer and Robot Vision, Volume II*. Addison-Wesley, Reading, MA, 1993.
- Hartley R. I. Estimation of relative camera positions for uncalibrated cameras. In *2nd European Conference on Computer Vision*, Santa Margherita Ligure, Italy, pages 579–587, Heidelberg, 1992. Springer Verlag.
- Hartley R. I. Self-calibration from multiple views with a rotating camera. In Eklundh J. O., editor, *3rd European Conference on Computer Vision*, Stockholm, Sweden, pages A:471–478, Berlin, 1994. Springer Verlag.
- Hartley R. I. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.
- Hartley R. I. and Zisserman A. *Multiple view geometry in computer vision*. Cambridge University, Cambridge, 2nd edition, 2003.
- Horaud R., Mohr R., Dornaika F., and Boufama B. The advantage of mounting a camera onto a robot arm. In *Proceedings of the Europe-China Workshop on Geometrical Modelling and Invariants for Computer Vision*, Xian, China, pages 206–213, 1995.
- Horn B. K. P. Shape from shading. In Winston P. H., editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.
- Horn B. K. P. Height and gradient from shading. *International Journal of Computer Vision*, 5(1):37–75, 1990.
- Horn B. K. P. *Shape from shading: A method for obtaining the shape of a smooth opaque object from one view*. Ph.D. thesis, Department of Electrical Engineering, MIT, Cambridge, MA, 1970.
- Howarth R. J. *Spatial representation and control for a surveillance system*. Ph.D. thesis, Department of Computer Science, Queen Mary and Westfield College, University of London, UK, 1994.
- Ikeuchi K. and Horn B. K. P. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17:141–184, 1981.
- Julesz B. Binocular depth perception of computer-generated patterns. *Bell Systems Technical Journal*, 39, 1990.
- Klette R., Koschan A., and Schlüns K. *Computer Vision—Räumliche Information aus digitalen Bildern*. Friedr. Vieweg & Sohn, Braunschweig, 1996.
- Klir G. J. *Facets of System Science*. Plenum Press, New York, 1991.
- Koenderink J. J. *Solid Shape*. MIT Press, Cambridge, MA, 1990.
- Landy M. S., Maloney L. T., and Pavel M., editors. *Exploratory Vision: The Active Eye*. Springer Series in Perception Engineering. Springer Verlag, New York, 1996.
- Longuet-Higgins H. C. A computer algorithm for reconstruction a scene from two projections. *Nature*, 293(10):133–135, 1981.
- Loop C. and Zhang Z. Computing rectifying homographies for stereo vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 125–131. IEEE Computer Society, 1999.
- Lütkepohl H. *Handbook of Matrices*. John Wiley & Sons, Chichester, England, 1996.
- Ma Y., Soatto S., Košecák J., and Sastry S. S. *An invitation to 3-D vision : from images to geometric models*. Interdisciplinary applied mathematic. Springer, New York, 2004.
- Marr D. *Vision—A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman, San Francisco, 1982.

- Marr D. and Hildreth E. Theory of edge detection. *Proceedings of the Royal Society, B* 207: 187–217, 1980.
- Marr D. and Poggio T. A. A computational theory of human stereo vision. *Proceedings of the Royal Society, B* 207:301–328, 1979.
- Matoušek M., Šára R., and Hlaváč V. Data-optimal rectification for fast and accurate stereo-vision. In Zhang D. and Pan Z., editors, *Proceedings of the Third International Conference on Image and Graphics*, pages 212–215, Los Alamitos, USA, 2004. IEEE Computer Society.
- Maybank S. J. and Faugeras O. D. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.
- Mohr R. Projective geometry and computer vision. In Chen C. H., Pau L. F., and Wang P. S. P., editors, *Handbook of Pattern Recognition and Computer Vision*, chapter 2.4, pages 369–393. World Scientific, Singapore, 1993.
- Newman T. S., Flynn P. J., and Jain A. K. Model-based classification of quadric surfaces. *CVGIP: Image Understanding*, 58(2):235–249, 1993.
- Nishihara H. K. Practical real-time imaging stereo matcher. *Optical Engineering*, 23(5):536–545, 1984.
- Oliensis J. Shape from shading as a partially well-constrained problem. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 54(2):163–183, 1991.
- Pajdla T. and Hlaváč V. Camera calibration and Euclidean reconstruction from known translations. Presented at the workshop *Computer Vision and Applied Geometry*, 1995.
- Pentland A. P. Local shading analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):170–187, 1984.
- Pentland A. P. and Bichsel M. Extracting shape from shading. In Young T. Y., editor, *Handbook of Pattern Recognition and Image Processing: Computer Vision*, pages 161–183, San Diego, 1994. Academic Press.
- Poggio T., Torre V., and Koch C. Computational vision and regularization theory. *Nature*, 317: 314–319, 1985.
- Pollard S. B., Mayhew J. E. W., and Frisby J. P. PMF: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1985.
- Pollefeys M., Koch R., and VanGool L. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the International Conference on Computer Vision*, pages 90–95, New Delhi, India, 1998. IEEE Computer Society, Narosa Publishing House.
- Press W. H., Teukolsky S. A., Vetterling W. T., and Flannery B. P. *Numerical Recipes in C*. Cambridge University Press, Cambridge, England, 2nd edition, 1992.
- Rindfleisch T. Photometric method form lunar topography. *Photogrammetric Engineering*, 32 (2):262–267, 1966.
- Sara R. *Local shading analysis via isophotes properties*. Ph.D. thesis, Department of System Sciences, Johannes Kepler University Linz, 1994.
- Sara R. Finding the largest unambiguous component of stereo matching. In Heyden A., Sparr G., Nielsen M., and Johansen P., editors, *Proceedings 7th European Conference on Computer Vision*, volume 3 of *Lecture Notes in Computer Science 2352*, pages 900–914, Berlin, Germany, May 2002. Springer.
- Sara R. Isophotes: The key to tractable local shading analysis. In Hlavac V. and Sara R., editors, *International Conference on Computer Analysis of Images and Patterns*, Prague, Czech Republic, pages 416–423, Heidelberg, 1995. Springer Verlag.

- Semple J. G. and Kneebone G. T. *Algebraic Projective Geometry*. Oxford University Press, London, 1963.
- Smutný V. Analysis of rainbow range finder errors. In Hlaváč V. and Pajdla T., editors, *1st Czech Pattern Recognition Workshop*, pages 59–66. Czech Pattern Recognition Society, CTU, Prague, 1993.
- Soucy M. and Laurendeau D. Surface modeling from dynamic integration of multiple range views. In *11th International Conference on Pattern Recognition*, The Hague, volume I, pages 449–452, Piscataway, NJ, 1992. IEEE.
- Svoboda T., Kybic J., and Hlavac V. *Image Processing, Analysis, and Machine Vision: A MATLAB Companion*. Thomson Engineering, 2008.
- Szeliski R. Fast surface interpolation using hierarchical basis functions. *Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528, June 1990.
- Tichonov A. N. and Arsenin V. Y. *Solution of ill-posed problems*. Winston and Wiley, Washington, DC, 1977.
- Triggs B., McLauchlan P., Hartley R., and Fitzgibbon A. Bundle adjustment – A modern synthesis. In Triggs W., Zisserman A., and Szeliski R., editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *LNCS*, pages 298–375. Springer Verlag, 2000.
- Tsai R. Y. An efficient and accurate camera calibration technique for 3D machine vision. In *CVPR '86: Computer Society Conference on Computer Vision and Pattern Recognition*, Miami Beach, FL, pages 364–374, 1986.
- Ullman S. *High-Level Vision: Object Recognition and Visual Cognition*. MIT Press, Cambridge, MA, 1996.
- Ullman S. and Basri R. Recognition by linear combination of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1005, 1991.
- Wechsler H. *Computational Vision*. Academic Press, London–San Diego, 1990.
- Werner T., Hersch R. D., and Hlaváč V. Rendering real-world objects using view interpolation. In *5th International Conference on Computer Vision*, Boston, USA, pages 957–962. IEEE Computer Society, 1995.
- Woodham R. J. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19:139–144, 1980.

Chapter 12

Use of 3D vision

In earlier (and later) chapters, we present a constructive approach to various aspects of image processing and vision that should allow readers to reproduce ideas and build systems of their own. Most of this chapter is somewhat different; 3D vision solves complex tasks having no settled and simple theory, and here we step aside and provide an overview of selected approaches, task formulations, applications and current research. We hope thereby that the reader will learn what the current state of 3D vision is; it is possible that the material herein will be at the limit of the abilities of the vision novice, but may form a useful primer for master's or Ph.D. courses.

12.1 Shape from X

Shape from X is a generic name for techniques that aim to extract shape from intensity images and other cues such as focus. Some of these methods estimate local surface orientation (e.g., surface normal) rather than absolute depth. If, in addition to this local knowledge, the depth of some particular point is known, the absolute depth of all other points can be computed by integrating the surface normals along a curve on a surface [Horn, 1986]. Several topics belonging to this category of methods have already been mentioned, i.e., *shape from stereo* (Sections 11.5, 11.6.1), *shape from shading* (Section 11.7.1), and *photometric stereo* (Section 11.7.2).

12.1.1 Shape from motion

Motion is a primary property exploited by human observers of the 3D world. The real world we see is dynamic in many respects, and the relative movement of objects in view, their translation and rotation relative to the observer, the motion of the observer relative to other static and moving objects all provide very strong clues to shape and depth—consider how just moving your head from side to side provides rich information from parallax effects. It should therefore come as no surprise to learn that attempts at shape extraction are able to make use of motion. Motion, and particularly lower-level

algorithms associated with its analysis, is considered in detail in Chapter 16, and in this section study is restricted to shape extraction alone.

A study of human analysis of motion is instructive and was conducted comprehensively in a computational context by Ullman [Ullman, 1979]. Exactly how we make deductions from moving scenes is far from clear, and several theories have come and gone in efforts to understand this matter—in particular, **Gestaltist** theories. Gestalt psychology was a revolutionary psychological paradigm proposed in Germany in the early twentieth century (‘Gestalt’ means ‘shape’ or ‘form’ in German). It claims that more complicated mental processes cannot be simply composed from the simpler ones, and questioned the causality of events. Its suggestion that groupings of observations are of primary importance was disproved, notably by an experiment of Ullman’s. On a computer screen, he simulated two coaxial cylinders of different radii rotating about their common axis in opposite directions. The view is perpendicular to the common axis; the cylinders were not drawn, but only randomly placed dots on their surfaces. Thus what is seen (on a per-point basis) is a large number of dots moving from left to right or right to left, at varying speeds. Exactly what speed and direction depends upon which cylinder surface a dot belongs to, and at what point of rotation it is—in fact, each individual dot executes simple harmonic motion about a point that is on a line that is the projection onto the image of the axis. The striking conclusion is that the human observer is in no doubt about the nature of the scene, despite the absence of surface clues and the complete absence of structure in any single frame from the sequence.

What we exploit are particular **constraints that assist in resolving the non-uniqueness of the interpretation** of a sequence of frames as a moving 3D scene. In fact, motion may be presented to us as widely spaced (in time) discrete frames, or as (pseudo-)continuous—that is, so many frames that changes between a given pair are imperceptible. We shall examine each case separately, each time using Ullman’s observation that the extraction of 3D information from moving scenes can be done as a two-phase process:

1. **Finding correspondences** or calculating the nature of the flow is a lower-level phase that operates on pixel arrays.
2. The **shape extraction** phase follows as a separate, higher-level process. This phase is examined here.

It is worth noting that researchers are not unanimous in the view that these two phases should be held separate, and approaches exist that are different from those discussed here [Negahdaripour and Horn, 1985].

Note that one approach to the analysis of motion is superficially similar to that of stereo vision—images that are relatively widely separated in time are taken, and correspondences between visible features made. The solution to this correspondence problem is considered in detail in Chapter 16 and Section 11.6.1. It is worth remarking here that resemblance to the stereo correspondence problem is deceptive since the scene may well contain any number of independently moving objects, which could mean that correlations may be strictly local. Two images are not of the same scene, but (more probably) of the same objects in different relative positions.

Searching for correspondence in motion analysis may be easier than when attempting it in stereo imaging. It is often possible to capture a dense sequence of images (i.e., the time separation between neighboring frames is small so that corresponding features

are very close, and the search for them almost trivial). Moreover, the position of the feature in the next frame can be predicted by estimating its trajectory using techniques similar to those of control theory. The Kalman filter and Condensation approaches (see Section 16.6.1) are common.

Rigidity, and the structure from motion theorem

For now, suppose that the correspondence problem has been solved, and that it remains to extract some shape information—that is, given that a collection of points has been identified in two different views, how might they be interpreted as 3D objects? As might be expected, the large number of possible interpretations is resolved by deploying a constraint; Ullman's success in this area was based on the psycho-physical observation that the human visual system seems to assume that objects are *rigid*. This rigidity constraint prompted the proof of an elegant **structure from motion theorem** saying that *three orthographic projections of four non-co-planar points have a unique 3D interpretation as belonging to one rigid body*. We shall proceed to outline the proof of this theorem, which is constructive and therefore permits the extraction of the appropriate geometry, given point correspondences in three frames from a motion sequence. In use, the theorem allows samples of four points to be taken from an image sequence—if they belong to the same (rigid) body, an interpretation is generated, but if they do not, the probability of there being a chance rigid interpretation turns out to be negligibly small, meaning that the algorithm is self-verifying in the sense that it generates only answers that are 'correct'. Thus if there are N points in the correspondence, we might test all selections of 4 points for rigid interpretation; some of these will be invalid, and others will group according to the rigid object to which they belong.

The theorem proof involves a re-phrasing of the problem to permit its definition as the solution of an equivalent problem in 3D geometry. Given three orthographic views of four points that have a rigid interpretation, the correspondence allows them to be labeled as O , A , B , and C in each image. First note that the body's motion may be decomposed into translational and rotational movement; the former gives the movement of a fixed point with respect to the observer, and the latter relative rotation of the body (for example, about the chosen fixed point). **Translational movement**, as far as it is recognizable, is easy to identify. All that can be resolved is movement perpendicular to the projection, and this is given by the translation (in 2D) of an arbitrarily chosen point, say O . Observe that motion parallel to the projection cannot be identified.

It remains to identify **rotational motion**; to do this we can assume that O is a fixed point, and seek to identify an interpretation of A , B and C as belonging to the same rigid body as O . Accordingly, we transform the problem to that of knowing three pairs of (2D) co-ordinates for A , B , and C with respect to a common origin O , each a different orthographic projection; what is now required is the (3D) directions of the projections.

Formally, suppose we have in 3D an origin O and three vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ corresponding to A , B , and C ; given projections of $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ onto three planes Π_1 , Π_2 , and Π_3 of unknown orientation, we require to reconstruct the 3D geometry of $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$. Now let the co-ordinate system of the plane Π_i be defined by vectors $\mathbf{x}_i$ and $\mathbf{y}_i$; that is, $\mathbf{x}_i$ and $\mathbf{y}_i$ are orthogonal 3D unit vectors lying in the plane Π_i . With respect to these systems, suppose that on plane Π_i the points' projections have co-ordinates (a_{xi}, a_{yi}) , (b_{xi}, b_{yi}) , (c_{xi}, c_{yi}) —these nine pairs are the input to the algorithm. Finally, let $\mathbf{u}_{ij}$ be a unit vector lying on the line defined by the intersection of planes Π_i and Π_j .

Elementary co-ordinate geometry gives

$$\begin{aligned} a_{xi} &= \mathbf{a} \cdot \mathbf{x}_i, & a_{yi} &= \mathbf{a} \cdot \mathbf{y}_i, \\ b_{xi} &= \mathbf{b} \cdot \mathbf{x}_i, & b_{yi} &= \mathbf{b} \cdot \mathbf{y}_i, \\ c_{xi} &= \mathbf{c} \cdot \mathbf{x}_i, & c_{yi} &= \mathbf{c} \cdot \mathbf{y}_i. \end{aligned} \quad (12.1)$$

Further, since $\mathbf{u}_{ij}$ lies on both Π_i and Π_j , there must exist scalars $\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}$ such that

$$\alpha_{ij}^2 + \beta_{ij}^2 = 1, \quad \gamma_{ij}^2 + \delta_{ij}^2 = 1, \quad (12.2)$$

and

$$\begin{aligned} \mathbf{u}_{ij} &= \alpha_{ij} \mathbf{x}_i + \beta_{ij} \mathbf{y}_i, \\ \mathbf{u}_{ij} &= \gamma_{ij} \mathbf{x}_j + \delta_{ij} \mathbf{y}_j, \end{aligned} \quad (12.3)$$

and hence

$$\alpha_{ij} \mathbf{x}_i + \beta_{ij} \mathbf{y}_i = \gamma_{ij} \mathbf{x}_j + \delta_{ij} \mathbf{y}_j. \quad (12.4)$$

We can take the scalar product of this equation with each of $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, and using equation (12.1) see that

$$\begin{aligned} \alpha_{ij} a_{xi} + \beta_{ij} a_{yi} &= \gamma_{ij} a_{xj} + \delta_{ij} a_{yj}, \\ \alpha_{ij} b_{xi} + \beta_{ij} b_{yi} &= \gamma_{ij} b_{xj} + \delta_{ij} b_{yj}, \\ \alpha_{ij} c_{xi} + \beta_{ij} c_{yi} &= \gamma_{ij} c_{xj} + \delta_{ij} c_{yj} \end{aligned} \quad (12.5)$$

—thus we have relations between unknowns $(\alpha, \beta, \gamma, \delta)$ in terms of known quantities $(a_x, a_y, \text{etc.})$.

It is easy to show that the equations (12.5) are linearly independent (this is where the fact that O , A , B , and C are not co-planar is used). Therefore, using the constraint of equation (12.2), it is possible to solve for $\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}$ —in fact, there are two possible solutions that differ in sign only.

This (findable) solution is important, as it means that we are able to express the vectors $\mathbf{u}_{ij}$ in terms of the co-ordinate basis vectors $\mathbf{x}_i, \mathbf{y}_i, \mathbf{x}_j$, and $\mathbf{y}_j$. To see why this is important, picture the three planes in 3D—they intersect at the common origin O and therefore define a tetrahedron; what interests us is the *relative* angles between the planes, and if the geometry of the tetrahedron can be recaptured, these angles are available. Note, though, that knowledge of $\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}$ allows calculation of the distances

$$\begin{aligned} d_1 &= |\mathbf{u}_{12} - \mathbf{u}_{13}|, \\ d_2 &= |\mathbf{u}_{12} - \mathbf{u}_{23}|, \\ d_3 &= |\mathbf{u}_{13} - \mathbf{u}_{23}|. \end{aligned} \quad (12.6)$$

For example

$$\begin{aligned} \mathbf{u}_{12} - \mathbf{u}_{13} &= (\alpha_{12} \mathbf{x}_1 + \beta_{12} \mathbf{y}_1) - (\alpha_{13} \mathbf{x}_1 + \beta_{13} \mathbf{y}_1) \\ &= (\alpha_{12} - \alpha_{13}) \mathbf{x}_1 + (\beta_{12} - \beta_{13}) \mathbf{y}_1, \end{aligned} \quad (12.7)$$

and hence

$$d_1 = (\alpha_{12} - \alpha_{13})^2 + (\beta_{12} - \beta_{13})^2. \quad (12.8)$$

since $\mathbf{x}_1$ and $\mathbf{y}_1$ are orthogonal. Now the tetrahedron formed by the three intersecting planes is defined by the origin O and a triangular base—we might consider the base given by the three points at unit distance from the origin. By construction, this triangle has sides d_1, d_2, d_3 , and we can thus reconstruct the required tetrahedron.

Determining the 3D structure is now possible by noting that a particular point lies at the intersection of the normals to any two of the planes drawn from the projections of the point concerned.

There is a complication in the proof not discussed here that occurs when one of the d_i is zero, and the tetrahedron is degenerate. It is possible to resolve this problem without difficulty [Ullman, 1979].

It is worth noting that Ullman's result is the best possible in the sense that unique reconstruction of a rigid body cannot be guaranteed with fewer than three projections of four points, or with three projections of fewer than four points. It should also be remembered that the result refers to *orthographic* projection when in general image projections are *perspective* (of which, of course, the orthographic projection is a special case). This turns out not to be a problem since a similar result is available for the perspective projection [Ullman, 1979]. In fact this is not necessary, since it is possible to approximate neighborhoods within a perspective projection by a number of different orthographic projections; thus in such a neighborhood, the theorem as outlined is valid. Interestingly, there seems to be evidence that the human visual system uses this sort of orthographic approximation in extracting shape information from motion.

This result is of particular value in **active vision** applications [Blake and Yuille, 1992; Aloimonos, 1993] such as a robot arm having a camera mounted upon it; when such a system finds itself unable to 'see' particular objects of interest, the arm will move for a different view, which will then need reconciling with earlier ones.

Shape from optical flow

The motion presented to human observers is not that considered in the previous section, but rather is continuous—the scene in view varies smoothly. The approach of considering widely spaced (in time) views is therefore a simplification, and it is natural to ask how to treat the limiting case of separate frames being temporally very close to each other—it is well known that, in fact, the human eye perceives continuous motion from relatively few frames per second (as illustrated by cinema film). Clearly the approach of making correspondences is no longer any use since corresponding points will be separated by infinitesimally small distances—it is the apparent velocity (direction and speed) of pixels that is of interest in the study of continuous motion. In a continuous sequence, we are therefore interested in the apparent movement of each pixel (x, y) which is given by the **optical flow field** $(dx/dt, dy/dt)$. In Chapter 16, optical flow is considered at length and an algorithm is described for its extraction from observation of changes in the intensity function (gray-levels); accordingly, in this section it is assumed that the flow field is available, and we ask how it may be used to extract shape in the form of surface orientation (in fact, optical flow is useful for deducing a number of motion properties, such as the nature of the translational or rotational movement—these points are considered in Chapter 16).

Determining shape from optical flow is mathematically non-trivial, and here an early simplification of the subject is presented as an illustration [Clocksin, 1980]. The simplification is in two parts:

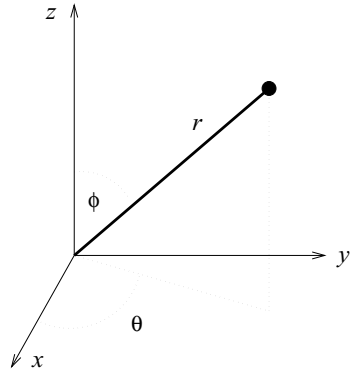


Figure 12.1: Definition of spherical polar co-ordinates.
© Cengage Learning 2015.

- Motion is due to the observer travelling in a straight line through a static landscape. Without loss of generality, suppose the motion is in the direction of the z axis of a viewer-centered co-ordinate system (i.e., the observer is positioned at the origin).
- Rather than being projected onto a 2D plane, the image is seen on the surface of a unit sphere, centered at the observer (a ‘spherical retina’). Points in 3D are represented in spherical polar rather than Cartesian co-ordinates—spherical polar co-ordinates (r, θ, φ) (see Figure 12.1) are related to (x, y, z) by the equations

$$r^2 = x^2 + y^2 + z^2, \quad (12.9)$$

$$y = x \tan \theta, \quad (12.10)$$

$$z = r \cos \varphi. \quad (12.11)$$

Since the image is spherical, we can specify co-ordinates as (θ, φ) pairs rather than (x, y) as usual, and the optical flow is then $(d\theta/dt, d\varphi/dt)$. Supposing the observer’s speed to be v (in the direction of the z axis), the motion of points in 3D is given by

$$\frac{dx}{dt} = 0, \quad \frac{dy}{dt} = 0, \quad \frac{dz}{dt} = -v. \quad (12.12)$$

Differentiating equation (12.9) with respect to t gives

$$\begin{aligned} 2r \frac{dr}{dt} &= 2x \frac{dx}{dt} + 2y \frac{dy}{dt} + 2z \frac{dz}{dt} \\ &= -2vz, \\ \frac{dr}{dt} &= -\frac{vz}{r} \\ &= -v \cos \varphi. \end{aligned} \quad (12.13)$$

Differentiating equation (12.10) with respect to t gives

$$\begin{aligned} \frac{dy}{dt} &= \tan \theta \frac{dx}{dt} + x \sec^2 \theta \frac{d\theta}{dt}, \\ 0 &= 0 + x \sec^2 \theta \frac{d\theta}{dt}, \end{aligned} \quad (12.14)$$

and hence

$$\frac{d\theta}{dt} = 0. \quad (12.15)$$

Differentiating equation (12.11) with respect to t gives

$$\frac{dz}{dt} = \cos \varphi \frac{dr}{dt} - r \sin \varphi \frac{d\varphi}{dt}$$

and hence, by equations (12.12) and (12.13)

$$-v = -v \cos^2 \varphi - r \sin \varphi \frac{d\varphi}{dt}$$

and so

$$\frac{d\varphi}{dt} = \frac{v(1 - \cos^2 \varphi)}{r \sin \varphi} = \frac{v \sin \varphi}{r}. \quad (12.16)$$

Equations (12.15) and (12.16) are important. The former says that, for this particular motion, the rate of change of θ is zero (θ is constant). More interestingly, the latter says that given the optical flow $d\varphi/dt$, then the distance r of a 3D point from the observer can be recaptured up to a scale factor v . In particular, if v is known, then r , and a complete depth map, can be deduced from the optical flow. The depth map allows a reconstruction of the 3D scene and hence characteristics of surfaces (smoothly varying areas of r) and of edges (discontinuities in r) will be available.

In the case that v is not known, it turns out that surface information is still available directly from the flow. In particular, suppose a point P lies on a smooth surface, which at P may be specified by the direction of a normal vector $\mathbf{n}$. Such a direction may be specified by two angles α and β , where α is the angle between $\mathbf{n}$ and a plane Π_1 defined by P and the z axis, and β is the angle between $\mathbf{n}$ and a plane Π_2 which passes through P and the origin and is perpendicular to Π_1 . Intuitively, it is clear that the rate of change of r with respect to θ and φ provides information about the direction of $\mathbf{n}$. Moderately straightforward co-ordinate geometry gives the relations

$$\tan \alpha = \frac{1}{r} \frac{\partial r}{\partial \varphi}, \quad \tan \beta = \frac{1}{r} \frac{\partial r}{\partial \theta}. \quad (12.17)$$

These equations depend upon a knowledge of r (the depth map), but it is possible to combine them with equation (12.16) to overcome this. For convenience, write $d\varphi/dt = \dot{\varphi}$; then, by equation (12.16)

$$r = \frac{v \sin \varphi}{\dot{\varphi}} \quad (12.18)$$

and so

$$\begin{aligned} \frac{\partial r}{\partial \varphi} &= v \frac{\dot{\varphi} \cos \varphi - \sin \varphi (\partial \dot{\varphi} / \partial \varphi)}{\dot{\varphi}^2}, \\ \frac{\partial r}{\partial \theta} &= -v \frac{\sin \varphi (\partial \dot{\varphi} / \partial \theta)}{\dot{\varphi}^2}. \end{aligned} \quad (12.19)$$

Substituting (12.18) and (12.19) into equations (12.17) gives

$$\begin{aligned} \tan \alpha &= \cot \varphi - \frac{1}{\dot{\varphi}} \frac{\partial \dot{\varphi}}{\partial \varphi}, \\ \tan \beta &= \frac{1}{\dot{\varphi}} \frac{\partial \dot{\varphi}}{\partial \theta}. \end{aligned} \quad (12.20)$$

Thus, given the flow $\dot{\varphi}$ (which we assume), the angles α and β are immediately available, regardless of S and without any need to determine the depth map given by r .

The original reference [Clocksin, 1980] provides full information on this derivation, and proceeds to describe how edge information may also be extracted from knowledge of the flow. It also includes some interesting discussion of psycho-physical considerations of human motion perception in the context of a computational theory.

12.1.2 Shape from texture

A further property of which there is clear psycho-physical evidence of human use to extract depth is texture [Marr, 1982]. To appreciate this, it is only necessary to consider a regularly patterned object viewed in 3D. Two effects would be apparent: The angle at which the surface is seen would cause a (perspective) distortion of the **texture primitive** (**texel**), and the relative size of the primitives would vary according to distance from the observer. Simple examples, shown in Figure 12.2, are sufficient to illustrate this. Much use can be made of texture in computer vision at various levels of abstraction, and Chapter 15 examines them in some detail. Here we look briefly at the use of textural properties to assist in the extraction of shape [Bajcsy and Lieberman, 1976; Kanatani and Chou, 1989].

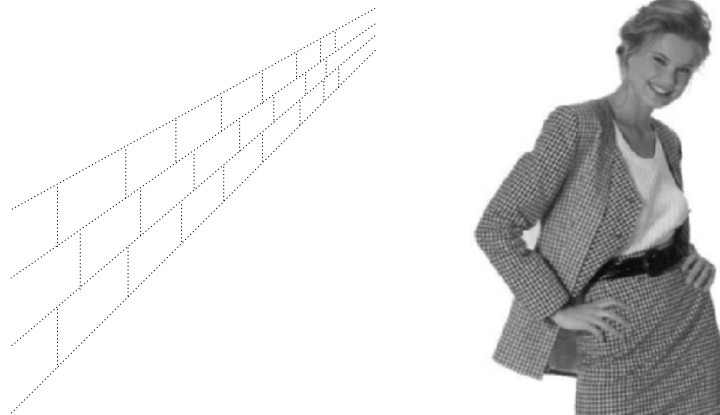


Figure 12.2: A simple texture pattern in 3D. The left side shows a vanishing brick wall and the right the shape of a body perceived from texture changes. © Cengage Learning 2015.

Considering a textured surface patterned with identical texels which have been recovered by lower-level processing, note that with respect to a viewer it has three properties at any point projected onto a retinal image: distance from the observer, **slant**; the angle at which the surface is sloping away from the viewer (the angle between the surface normal and the line of sight); and **tilt**, the direction in which the slant takes place. Attempts to re-capture some of this information is based on the **texture gradient**—that is, the direction of maximum rate of change of the perceived size of the texels, and a scalar measurement of this rate. One approach [Bajcsy and Lieberman, 1976] assumes a uniform texel size.

If the texture is particularly simple, the shape of the perceived texels will reveal surface orientation information. For example, if a plane is marked with identical circles, they will be seen in an image as ellipses (see Figure 12.3). The eccentricity of the ellipses provides information about the slant, while the orientation of the ellipse axes indicates the tilt [Stevens, 1979]. There is evidence to suggest [Stevens, 1979] that the human viewer

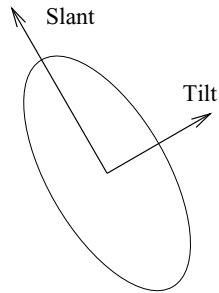


Figure 12.3: Slant and tilt are revealed by texel properties. © Cengage Learning 2015.

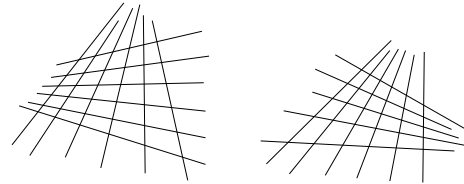


Figure 12.4: Tilt affects the appearance of texture. © Cengage Learning 2015.

uses texture gradient as a primary clue in the extraction of tilt and relative distance, but that slant is inferred by processing based on estimates of the other two parameters. Tilt is indicated by the direction of the texture gradient (see Figure 12.4), while the apparent size of objects decreases as the reciprocal of their distance from the viewer.

Large-scale texture effects can provide information about large-scale scene geometry; in particular, strong linear effects will indicate ‘vanishing points’ that may be joined to give the scene horizon. Note that it is not necessary in fact for the image to contain large numbers of long straight lines for this conclusion to be available, since often such lines can be inferred by a number of segments whose co-linearity may be deduced by, for example, a Hough transform. Such co-linearity may well be a property of urban scenes in which rectangular objects, some large, can predominate.

Texture has many interpretations and there is a correspondingly large number of attempts to exploit it in shape extraction—a useful grounding may be found in [Witkin, 1981]. A multiple-scale approach was used in [Blostein and Ahuja, 1989], while [Aloimonos and Swain, 1985] give an interesting approach in which ‘shape from texture’ is shown to be equivalent to ‘shape from shading’, thereby allowing the use of established results in surface parameter extraction from shading. Texture is usually used as an additional or complementary feature, augmenting another, stronger clue in shape extraction.

12.1.3 Other shape from X techniques

Shape from focus/de-focus techniques are based on the fact that lenses have finite depth of field, and only objects at the correct distance are in focus; others are blurred in proportion to their distance. Two main approaches can be distinguished:

Shape from focus measures depth in one location in an active manner; this technique is used in 3D measuring machines in mechanical engineering. The object to be measured is fixed on a motorized table that moves along x, y, z axes. A small

portion of the surface is observed by a camera through a microscopic lens, and if the surface patch in view (given by a small image window) is in focus, then the image has the maximal number of high frequencies; this qualitative information about focus serves as feedback to the z -axis servo-motor. The image is put into focus and x, y, z co-ordinates read from the motorized table. If the depth of all points in the image are to be measured, a large number of images is captured by displacing the sensor in small increments from the scene, and the image of maximum focus is detected for each image point [Krotkov, 1987; Nayar and Nakagawa, 1994].

Shape from de-focus typically estimates depth using two input images captured at different depths. The relative depth of the whole scene can be reconstructed from image blur. The image is modeled as a convolution of the image with a proper point spread function (see Section 3.1.2); the function is either known from capturing setup parameters or estimated, for example by observing a sharp depth step in the image. The depth reconstruction, which is an ill-posed problem [Pentland, 1987], is performed by local frequency analysis. Depth from de-focus shares an inherent problem with shape from stereo and shape from motion, in that it requires the scene to be covered by a fine texture. A real-time (30Hz) depth from de-focus sensor has been built [Nayar et al., 1996]: The device uses active illumination by texture and analyzes relative blur in two images captured at two different depths. The derivation of the illumination pattern and depth estimation is posed as an optimization problem in the Fourier domain.

Shape from vergence uses two cameras fixed on a common rod. Using two servo-mechanisms, the cameras can change the direction of their optical axes (verge) in the plane containing a line segment joining their optical centers. Such devices are called **stereo heads**; see Figure 12.5. The aim of shape from vergence is to ease the correspondence problem for estimating depth [Krotkov and Bajcsy, 1993]; vergence is used to align individual feature points in both left and right images.

Shape from contour aims to describe a 3D shape from contours seen from one or more view directions. Objects with smooth bounding surfaces are quite difficult to analyze.

Following terminology given in [Ullman, 1996], assume the object is observed from some view point. The set of all points on the object surface where surface normal is perpendicular to the observer's visual ray is called a **rim** [Koenderink, 1990]. Note that



Figure 12.5: An example of a stereo head.
© Cengage Learning 2015.

in general the rim is not a planar curve. Assuming orthographic projection, the rim points generate a **silhouette** of an object in the image. Silhouettes can be easily and reliably captured if back-light illumination is used, although there is possible complication in the special case in which two distinct rim points project to a single image point.

The most general approach considers contours as silhouettes plus images of the salient curves on the surface, e.g., those corresponding to surface curvature discontinuities. The latter are often found using an edge detector from an intensity image. The trouble is that this process is often not robust enough; the simpler—and more common—approach explores silhouettes as contours. In Figure 12.6, silhouette and surface discontinuity are shown on an image of an apricot.

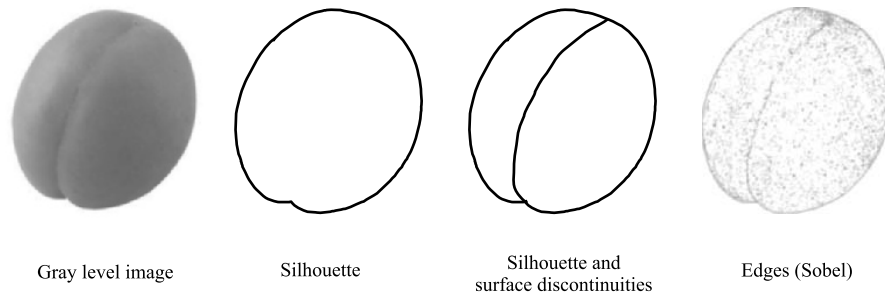


Figure 12.6: Apricot; contour as silhouette or as silhouette plus surface discontinuities, insufficient result of Sobel edge detector © Cengage Learning 2015.

The inherent difficulty in *shape from contour* comes from the loss of information in projecting 3D to 2D. We know that this projection is not invertible—this is illustrated in Figure 12.7, in which both a sphere and an ellipsoid project to the same ellipse. Humans are surprisingly successful at perceiving clear 3D shapes from contours, and it seems that tremendous background knowledge is used to assist. Understanding this human ability is one of the major challenges for computer vision. Contours are used as constraints on the shape they represent, and the aim is to reduce the number of possible interpretations. A fuller coverage of these approaches is in [Nevatia et al., 1994].

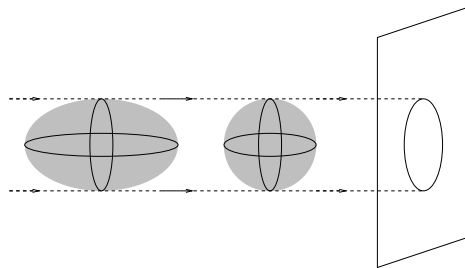


Figure 12.7: Ambiguity of the shape from contour task. © Cengage Learning 2015.

12.2 Full 3D objects

12.2.1 3D objects, models, and related issues

The notion of a **3D object** allows us to consider a 3D volume as a part of the entire 3D world. This volume has a particular interpretation (semantics, purpose) for the task in hand. Such an approach accords with the way general systems theory [Klir, 1991] treats complex phenomena, in which objects are separated from uninteresting background. Thus far, we have treated geometric (Section 11.2) and radiometric (Section 3.4.4) techniques that provide intermediate 3D cues, and it was implicitly assumed that such cues help to understand the nature of a 3D object.

Shape is another informal concept that humans typically connect with a 3D object. Recall the understanding we have when thinking of the shape of a mountain, or a vase or a cup. Computer vision aims at scientific methods for 3D object description, but there are no mathematical tools yet available to express shape in its general sense. The reader interested in abstract aspects of shape might consult texts on the shapes of solid objects [Koenderink, 1990]. Here, however, we shall not consider 3D shape issues in their full abstract sense. Instead, the simple geometrical approach treating parts of 3D objects locally as simple volumetric or surface primitives is used. Curvilinear surfaces with no restriction on surface shape are called **free-form surfaces**.

Roughly speaking, the 3D vision task distinguishes two classes of approach:

1. **Reconstruction** of the 3D object model or representation from real-world measurements with the aim of estimating a continuous function representing the surface.
2. **Recognition** of an instance of a 3D object in the scene. It is assumed that object classes are known in advance, and that they are represented by a suitable 3D model.

The reconstruction and recognition tasks use different representations of 3D objects. Recognition may use approaches that distinguish well between distinct classes, but do not characterize an object as a whole.

Humans meet and recognize often **deformable objects** that change their shape [Terzopoulos et al., 1988; Terzopoulos and Fleischer, 1988], an advanced topic that is too large to consider in this book.

Computer vision as well as computer graphics use **3D models** to encapsulate the shape of an 3D object. 3D models serve in computer graphics to generate detailed surface descriptions used to render realistic 2D images. In computer vision, the model is used either for reconstruction (copying, displaying an object from a different viewpoint, modifying an object slightly during animation) or for recognition purposes, where features are used that distinguish objects from different classes. There are two main classes of models: volumetric and surface. **Volumetric models** represent the ‘inside’ of a 3D object explicitly, while **surface models** use only object surfaces, as most vision-based measuring techniques can only see the surface of a non-transparent solid.

3D models make a transition towards an *object-centered* co-ordinate system, allowing object descriptions to be viewer independent. This is the most difficult phase within Marr’s paradigm. Successful implementation is remote, especially compared to the success seen with the derivation of the primal and 2.5D sketches. Unlike earlier stages, there is little physiological guidance that can be used to design algorithms, since this level of human vision is not well understood. Marr observes that the target co-ordinate system(s)

should be modular in the sense that each ‘object’ should be treated differently, rather than employing one global co-ordinate system—this prevents having to consider the orientation of model components with respect to the whole. Representations based on an object’s ‘natural’ axes, derived from symmetries, or the orientation of stick features, are likely to be of greater use.

3D models of objects are common in other areas besides computer vision, notably computer-aided design (CAD) and computer graphics, where image synthesis is required—that is, an exact (2D) pictorial representation of some modeled 3D object. Use of an object representation which matches the representation generated by CAD systems has been an active research area for years, with substantial promise for industrial model-based vision. Progress in this area is presented in [Bowyer, 1992], with papers devoted to CAD-based models applied to pose estimation [Kriegman, 1992; Ponce et al., 1992; Seales and Dyer, 1992], 3D specular object recognition [Sato et al., 1992], and invariant feature extraction from range data [Flynn and Jain, 1992].

Various representation schemes exist, with different properties. A representation is called **complete** if two different objects cannot correspond to the same model, so a particular model is unambiguous. A representation is called **unique** if an object cannot correspond to two different models. Most 3D representation methods sacrifice either the completeness or the uniqueness property. Commercial CAD systems frequently sacrifice uniqueness; different design methodologies may produce the same object. Some solid modelers maintain multiple representations of objects in order to offer flexibility in design.

Due to self-occlusion of objects and to triangulation-based measuring methods, most vision-based measuring sensors inherently produce only partial 3D descriptions of objects. A fusion of several such measurements from different viewpoints is needed to obtain the shape of an object entirely. An ideal 3D sensor would provide a set of 3D uniformly sampled points on the surface together with their relation to neighboring points.

12.2.2 Line labeling

Early attempts to develop 3D vision systems tried to reconstruct a full 3D representation from a single, fully segmented view of a scene. The step between the dimensions was made by assuming that all objects in the scene had planar faces (see Figure 12.8), and that three faces met at each vertex. A perfect segmentation then provides straight-edged regions, and in general three of these will meet at a vertex. The idea was that this constraint was sufficient to permit a single 2D view to permit unambiguous reconstruction of a polyhedron. For obvious reasons, this is sometimes called a **blocks world** approach.

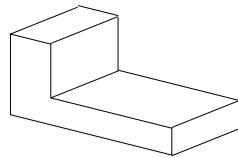


Figure 12.8: An example blocks world object. © Cengage Learning 2015.

The approach is clearly unrealistic for two reasons: First, the requirement for a perfect segmentation is unlikely to be met except in the most contrived situations; it is assumed that all edges are found, they are all linked into complete straight boundaries, and spurious evidence is filtered out. Second, there is a very limited number of circumstances in which objects do consist strictly of planar faces. It is perhaps possible that

industrial applications exist where both conditions might be met by constraining the objects, and providing lighting of a quality that permits full segmentation.

The idea was pioneered some time ago by Roberts [Roberts, 1965], who made significant progress, especially considering the time at which the work was done. Independently, other researchers built on these ideas to develop what is now a very well known **line labeling** algorithm [Clowes, 1971; Huffman, 1971]. Mindful of the limitations of the blocks world approach, research into 3D vision has left these ideas behind and they are now of historical interest only. What follows is an overview of how line labeling works, but it is instructive in that first, it illustrates how the 3D reconstruction task may be naively approached, and second, it is good example of **constraint propagation** (see Chapter 10) in action. The algorithm rests on observing that, since each 3D vertex is a meeting of exactly three planar faces, only four types of junction may appear in any 2D scene (see Figure 12.9). In the 3D world, an edge may be concave or convex, and in its 2D projection the three faces meeting at a vertex may be visible or occluded. These finite possibilities permit an exhaustive listing of interpretations of a 2D vertex as a 3D vertex—there are in fact 22 of them [Clowes, 1971].

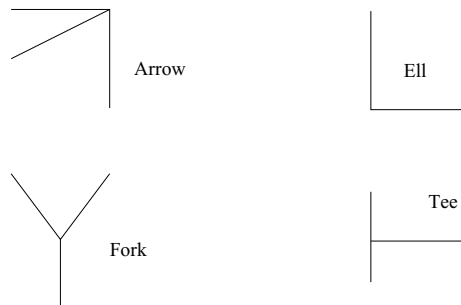


Figure 12.9: The four possible 2D junctions.
© Cengage Learning 2015.

The problem now reduces to deriving a mutually consistent set of vertex labels; this may be done by employing constraints such as an edge interpretation (convex or concave) being the same at both ends, and that circumnavigating a region provides a coherent 3D surface interpretation. At a high level, the algorithm is as follows.

Algorithm 12.1: Line labeling

1. Extract a complete and accurate segmentation of the 2D scene projection into polygons.
2. Determine the set of possible 3D interpretations for each 2D vertex from a pre-computed exhaustive list.
3. Determine ‘edge-wise’ coherent labellings of vertices by enforcing either concave or convex interpretations to each end of an edge.
4. Deduce an overall interpretation by requiring a circumnavigation of a region to have a coherent 3D interpretation.

Line labeling is able to detect ‘impossible’ objects such as that shown in Figure 12.10a, since it would not pass the final stage of the informally described algorithm; it would not, however, register Figure 12.10b, which defies a 3D interpretation along its upper front horizontal edge, as impossible. It is also unable, in the simple form described, to cope with ‘accidental’ junctions which are the meeting of four or more lines (caused by chance occlusion), although these could be analyzed as special cases.

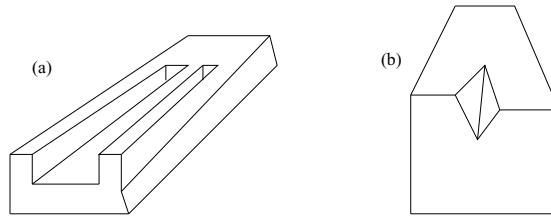


Figure 12.10: Impossible blocks world figures. © Cengage Learning 2015.

This simple approach received a lot of attention, and was extended to consider solids whose vertices may have more than three faces meeting at a vertex (such as square-based pyramids), and scenes in which regions might represent shadows of solids [Waltz, 1975]. Interestingly, while the number of possible junction interpretations increases enormously, the constraint satisfaction that is required for admissible candidate interpretations prevents the resulting algorithms becoming unworkable. More recent attempts at line label interpretations may be found in [Sugihara, 1986; Malik and Maydan, 1989; Shomar and Young, 1994]. In general, however, line labeling exists as an interesting historical idea. It is clear that its results are limited, and fraught with problems in overcoming ‘special cases’.

12.2.3 Volumetric representation, direct measurements

An object is placed in some reference co-ordinate system and its volume is subdivided into small volume elements called **voxels**—it is usual for these to be cubes. The most straightforward representation of voxel-based volumetric models is the **3D occupancy grid**, which is implemented as a 3D Boolean array. Each voxel is indexed by its x , y , z co-ordinates; if the object is present in a particular space location the voxel has value 1, and otherwise 0. Creating such a voxel-based model is an instance of discretization with similar rules to those for 2D images; for example, the Shannon sampling theorem (see Section 3.2.5) applies. An example of a voxelized toroid is shown in Figure 12.11. One way of obtaining a voxel-based volumetric model is to synthesize it using a geometric modeler, i.e., a computer graphics program. This permits composite objects to be assembled from some number of basic solids such as cubes, cylinders, and spheres.

Another possibility is the case in which a volumetric model is to be created from an existing real object. A simple measuring technique has been used in mechanical engineering for a long time. The object is fixed to a **measuring machine**, and an absolute co-ordinate system is attached to it. Points on the object surface are touched by a **measuring needle** which provides 3D co-ordinates; see Figure 12.12. The precision depends on the machine and the size of the objects, typically ± 5 micrometers.

In simpler machines, navigation of the needle on the surface is performed by a human operator; x , y , z co-ordinates are recorded automatically. Such a surface representation can easily be converted it into volumetric representation.

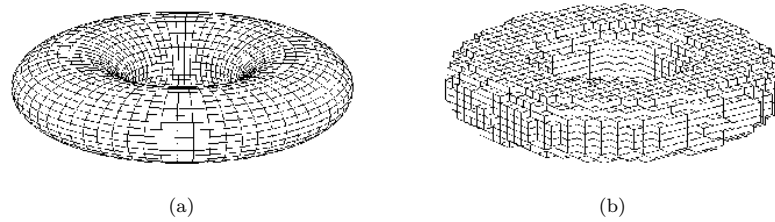


Figure 12.11: Voxelization (discretization) in 3D. (a) Continuous surface. (b) Voxelized image consisting of cubes of the same size. © Cengage Learning 2015.



Figure 12.12: An example of a fully motorized 3D measuring machine. Manufacturer, Mitutoyo, Inc., Japan. © Cengage Learning 2015.

Besides precision testing in machining or other mechanical engineering applications, this measuring technology may be used if the object is first created from clay by a designer. If computer-aided design (CAD) is to be brought into play, the 3D co-ordinates of the object are needed. An example is the automotive industry, where a clay model of a car body may be created at the scale 1:1. Actually only one half of the model is produced, as the car body is largely symmetric along the elongated axis. Such a model is measured on the 3D point measuring machine; as there are many points to be measured, the probe navigates semi-automatically. The points are organized into strips that cover the whole surface, and the probe has a proximity sensor that automatically stops on the surface or near to it. The probe is either a needle equipped with a force sensor or a laser probe performing the same measurement but stopping at a fixed and precise distance from the surface, e.g., 3 millimeters.

Another 3D measurement technique, **computed tomography**, looks inside the object and thus yields more detailed information than the binary occupancy grid discussed so far. Tomography yields a mass density in a 2D planar slice of the object. If 3D volumetric information is required, such slices are stacked on top of one another. The resulting 3D sample space consists of voxels, the values of which are mass densities addressed by the x , y , z co-ordinates. Computed tomography is used widely in medical imaging.

12.2.4 Volumetric modeling strategies

Constructive Solid Geometry

The principal idea of Constructive Solid Geometry (CSG), which has found some success is to construct 3D bodies from a selection of solid primitives. Popularly, these primitives are a cuboid, a cylinder, a sphere, a cone, and a ‘half-space’—the cylinder and cone are considered to be infinite. They are scaled, positioned, and combined by union, intersection, and difference; thus a finite cone is formed by intersecting an infinite cone with an appropriately positioned half-space. A CSG model is stored as a tree, with leaf nodes representing the primitive solid and edges enforcing precedence among the set theoretical operations. The versatility of such a simply stated scheme is surprising. CSG models define properties such as object volume unambiguously, but suffer the drawback of being non-unique. For example, the solid illustrated in Figure 12.13—a mug—may be formed by the union of the cylinder with a hole and a handle. The cylinder with the hole is obtained from a full cylinder by subtracting (in the set sense) a smaller cylinder. It is not easy to model ‘natural’ shapes (a human head, for instance) with CSG. A more serious drawback is that it is not straightforward to recover surfaces given a CSG description; such a procedure is computationally very expensive.

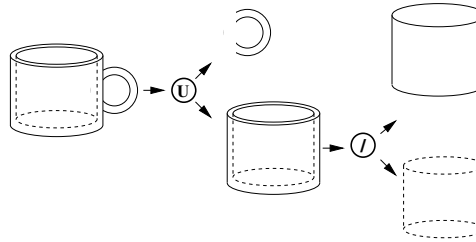


Figure 12.13: CSG representation of a 3D object—a mug. © Cengage Learning 2015.

Super-quadrics

Super-quadrics are geometric bodies that can be understood as a generalization of basic quadric solids, introduced in computer graphics [Barr, 1981]. Super-ellipsoids are instances of super-quadrics used in computer vision.

The implicit equation for a super-ellipsoid is

$$\left[\left(\frac{x}{a_1} \right)^{(2/\varepsilon_{\text{vert}})} + \left(\frac{y}{a_2} \right)^{(2/\varepsilon_{\text{vert}})} \right]^{(\varepsilon_{\text{hori}}/\varepsilon_{\text{vert}})} + \left(\frac{z}{a_3} \right)^{(2/\varepsilon_{\text{vert}})} = 1, \quad (12.21)$$

where a_1 , a_2 , and a_3 define the super-quadric size in the x , y , and z directions, respectively. $\varepsilon_{\text{vert}}$ is the squareness parameter in the latitude plane and $\varepsilon_{\text{hori}}$ is the squareness parameter in the longitude plane. The squareness values used in respective planes are 0 (i.e., square) $\leq \varepsilon \leq 2$ (i.e., deltoid), as only those are convex bodies. If squareness parameters are greater than 2, the body changes to a cross-like shape. Figure 12.14 illustrates how squareness parameters influence super-ellipsoid shape.

Super-quadric fitting to range images is described in [Solina and Bajcsy, 1990; Leonardis et al., 1997], and the construction of full 3D model range images taken from several

views using super-quadrics is shown in [Jaklič, 1997]. Super-quadric volumetric primitives can be deformed by bending, twisting, and tapering, and Boolean combinations of simple entities can be used to represent more complicated shapes [Terzopoulos and Metaxas, 1991].

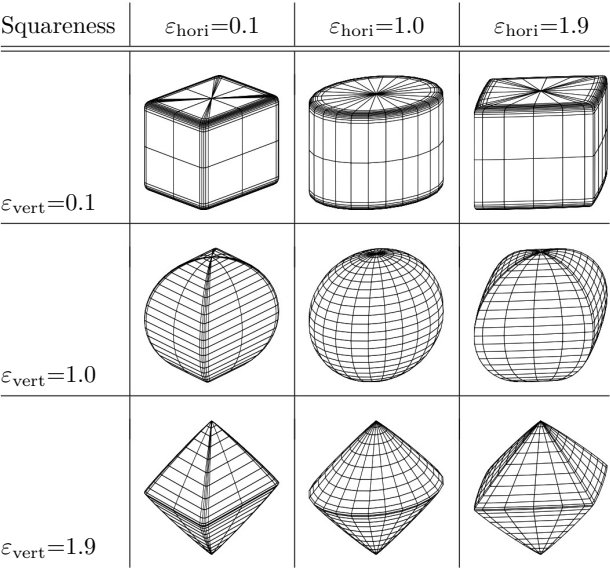


Figure 12.14: Super-ellipses.
© Cengage Learning 2015.

Generalized cylinders

Generalized cylinders, or **generalized cones**, are often also called **sweep representations**. Recall that a cylinder may be defined as the surface swept out by a circle whose center is traveling along a straight line (spine) normal to the circle’s plane. We can generalize this idea in a number of ways—we may permit any closed curve to be ‘pulled along’ any line in 3-space. We may even permit the closed curve to adjust as it travels in accordance with some function, so a cone is defined by a circle whose radius changes linearly with distance traveled, moving along a straight line. Further, the closed curve section need not contain the spine. Usually it is assumed that the curve is perpendicular to the spine curve at each point. In some cases this constraint is released. Figure 12.15 illustrates two simple generalized cylinders.

These generalized cones turn out to be very good at representing some classes of solid body [Binford, 1971; Soroka and Bajcsy, 1978]. The advantage of symmetrical

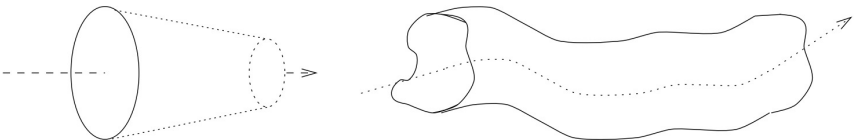


Figure 12.15: Solids represented as generalized cylinders. © Cengage Learning 2015.

volumetric primitives, such as generalized cylinders and super-quadrics, is their ability to capture common symmetries and represent certain shapes with few parameters. They are, however, ill-suited for modeling many natural objects that do not have the set of regularities incorporated into the primitives. An influential early vision system called ACRONYM [Brooks et al., 1979] used generalized cones as its modeling scheme.

There is a modification of the sweep representation called a **skeleton representation**, which stores only the spines of the objects [Besl and Jain, 1985].

12.2.5 Surface modeling strategies

A solid object can be represented by surfaces bounding it; such a description can vary from simple triangular patches to visually appealing structures such as non-uniform rational B-splines (NURBS) popular in geometric modeling. Computer vision solves two main problems with surfaces: First, reconstruction creates surface description from sparse depth measurements that are typically corrupted by outliers; second, segmentation aims to classify surface or surface patches into surface types.

Boundary representations (B-reps) can be viewed conceptually as a triple:

- A set of surfaces of the object.
- A set of space curves representing intersections between the surfaces.
- A graph describing the surface connectivity.

B-reps are an appealing and intuitively natural way of representing 3D bodies in that they consist of an explicit list of the bodies' faces. In the simplest case, 'faces' are taken to be planar, so bodies are always **polyhedral**, and we are dealing the whole time with piecewise planar surfaces. A useful side effect of this scheme is that properties such as surface area and solid volume are well defined. The simplest B-rep scheme would model everything with the simplest possible 2D polygon, the triangle. By taking small enough primitives quite satisfactory representations of complex objects can be achieved, and it is an obvious generalization to consider polygons with more edges than three.

Triangulation of irregular data points (e.g., a 3D point cloud obtained from a range scanner) is an example of an interpolation method. The best-known technique is called **Delaunay triangulation**, which can be defined in two, three, or more space dimensions. Delaunay triangulation is dual to the Voronoi diagram. We assume that the Euclidean distance between data points is known; then points that are closer to each other than to other points are connected. Let $d(P, Q)$ be the Euclidean distance between points P and Q , and S be the set of points $S = \{M_1, \dots, M_n\}$. A Voronoi diagram on the set S is a set of convex polyhedra that covers the whole space. The polyhedron V_i consists of all points that are closer to the point M_i than to other points of S

$$V_i = \{p; d(p, M_i) \leq d(p, M_j) \text{ for all } j = 1, 2, \dots, n\}. \quad (12.22)$$

An algorithm to compute Delaunay triangulation can be found in [Preparata and Shamos, 1985]. A problem with Delaunay triangulation is that it triangulates the convex hull of the point set; constrained Delaunay triangulation [Faugeras, 1993] can be a solution.

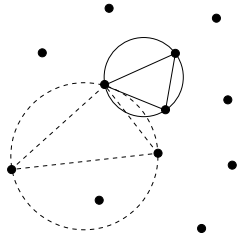


Figure 12.16: 2D Delaunay triangulation. The solid triangle belongs to the Delaunay triangulation but the dotted one does not, as its circumcircle contains an additional point. © Cengage Learning 2015.

We shall illustrate the idea of Delaunay triangulation on the simplest case of a 2D planar point set (see Figure 12.16). The task is to find triangles that cover all data points in such a way that the circumcircle of any one triangle contains only the three points that are vertices of that particular triangle. The triangulation has the following properties:

- The boundary of the set of points covered by triangles corresponds to the convex hull of the point set.
- The incremental algorithm constructing a triangulation of N points has expected time complexity $\mathcal{O}(N \log N)$ [Gubais et al., 1992].
- The 2D Delaunay triangulation algorithm provides a unique solution if no more than three points lie on one circle.

A drawback with polyhedral or triangulated B-reps is that the concept of ‘face’ may not be well defined. A face should have no ‘dangling’ edges, and the union of a body’s faces should be its boundary. Unfortunately, the real world is not cooperative, and many (simple) bodies exist in which face boundaries are not well defined.

The next step in generalizing descriptions of surface patches is the **quadric surface model**. Quadric surfaces are defined using second-degree polynomials in three co-ordinates x, y, z . In implicit form, the equation has up to 10 coefficients and represents hyperboloids, ellipsoids, paraboloids, and cylinders

$$\sum_{i,j,k=0\dots 2} a_{ijk} x^i y^j z^k = 0. \quad (12.23)$$

More complicated objects may be created from quadric surface patches. **Parametric bi-cubic surfaces** defined by bivariate cubic polynomials are used in CAD systems; the commonly used Bézier surfaces fall into this category. These surfaces have the advantage that surface patches can be smoothly joined along the intersection curves, and undesirable curvature discontinuity artifacts are thus avoided. Such an approach permits much greater flexibility in the description, but it becomes important to restrict the number of possible face edges in order to limit the complexity of the computations involved.

12.2.6 Registering surface patches and their fusion to get a full 3D model

A **range image** represents distance measurements from an observer to an object; it yields a partial 3D description of the surface from one view only. It may be visualized as a fresco made by a sculptor—shape information from different views, e.g., from the

other side of the object, is not available. Techniques for range image acquisition have been mentioned in Section 11.6.2.

Several range images are needed to capture the whole surface of an object. Each image yields a point cloud in the co-ordinates related to the range sensor, and successive images are taken in such a way that neighboring views overlap slightly, providing information for later fusion of partial range measurements into one global, object-centered, co-ordinate system.

A fusion of partial surface descriptions into global, object-centered co-ordinates implies known geometric transformations between the object and the sensor. The process depends on the data representing one view, e.g., from simple point clouds, triangulated surfaces, to parametric models as quadric patches.

Range image registration finds a rigid geometric transformation between two range images of the same object captured from two different viewpoints. The recovery can be based either on explicit knowledge of sensor positions, e.g., if it is held in a precise robot arm, or on geometric features measured from the overlapped parts of range data. Typically, both sources of information are used; an initial estimate of the appropriate geometric transformation can be provided by image feature correspondence, range image sensor data, an object manipulation device, or in many cases by a human operator.

This reconstruction task has been approached by several research groups, and many partial solutions have been proposed, e.g., [Hoppe et al., 1992; Higuchi et al., 1995; Uray, 1997]. We present here one of the possible approaches to the task. The method automates the construction of a 3D model of a 3D free-form object from a set of range images as follows.

1. The object is placed on a turntable and a **set of range images from different viewpoints is measured** by a structured-light (laser-plane) range finder.
2. A triangulated surface is constructed over the range images.
3. Large data sets are reduced by **decimation** of triangular meshes in each view.
4. Surfaces are registered into a common object-centered co-ordinate system and outliers in measurements are removed.
5. A full 3D model of the object is reconstructed by a surface fusion process.

Measurement from a laser-plane range finder provides a natural connectivity relation of points along stripes. Their parameterization allows easy construction of a 4-connected mesh following parametric curves just by connecting points found in the neighboring rows of the image and the points in neighboring scans with the same image row co-ordinate. The parameterization obtained of the measured surface is shown in Figure 12.17. The assumption of surface continuity is implemented as a restriction on the distance of neighboring points; only neighbors closer than a pre-defined ε are considered to lie on one surface next to each other. Points with no close neighbors are assumed to be outliers and are removed from data.

A 4-connected mesh cannot represent all objects; e.g., a sphere cannot be covered by a four-sided polygon. By splitting each polygon by an edge, a **triangulation** of the surface, which is able to represent any surface, is easily obtained. A polygon may be split two ways; it is preferable to choose the shortest edge because this results in triangles with larger inner angles.

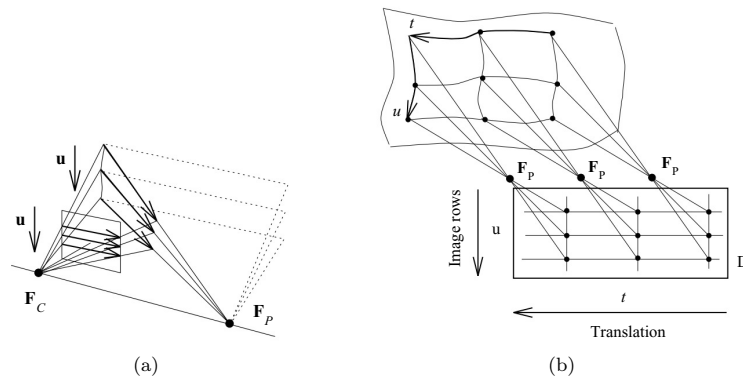


Figure 12.17: Surface parameterization is composed from the projected laser ray and translation.
© Cengage Learning 2015.

Often, we wish to reduce the number of triangles representing the surface in areas where curvature is low [Soucy and Laurendeau, 1996]. Data reduction is particularly useful for the registration of neighboring views, since it has worst-case complexity $\mathcal{O}(N^2)$ in the number of points. We formulate the task as a search for the best approximation of a triangulated surface by another triangulated surface that is close to the vertices of the original mesh [Hoppe et al., 1992]. For instance, we might look for the closest triangulated surface with maximally n triangles, or we might want simultaneously to minimize n and a residual error to get a consensus between the precision and space costs using the minimum description length principle (MDL) [Rissanen, 1989]. The surface triangulation procedure and node decimation is demonstrated on the synthetic pattern in Figures 12.18 and 12.19. Decimation of a triangulated surface from a real range image is shown in Figure 12.20.

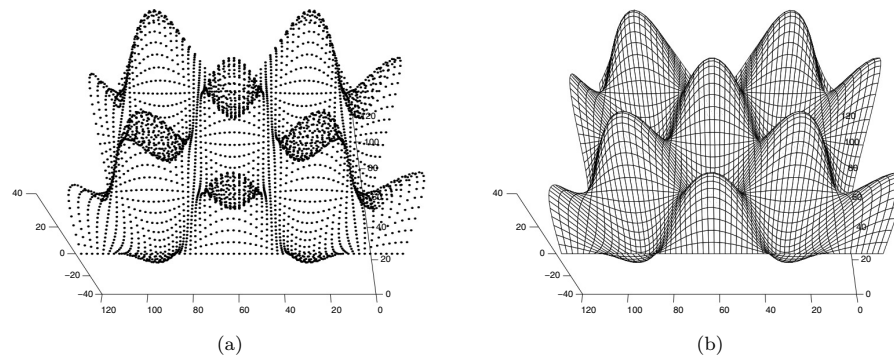


Figure 12.18: Surface construction and its decimation on a synthetic sinusoidal pattern.
(a) Point cloud. (b) 4-connected mesh. *Courtesy of T. Pajdla, D. Večerka, Czech Technical University, Prague.*

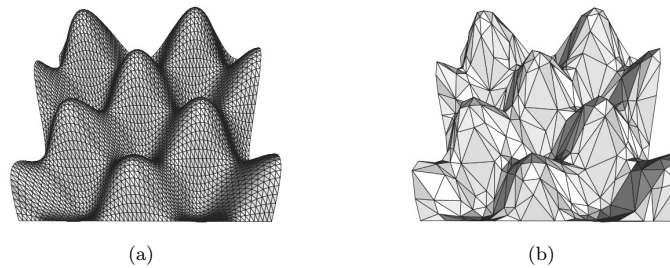


Figure 12.19: Surface construction and its decimation on a synthetic sinusoidal pattern. (a) Triangulated surface. (b) Surface after decimation of a number of triangles. *Courtesy of T. Pajdla, D. Večerka, Czech Technical University, Prague.*

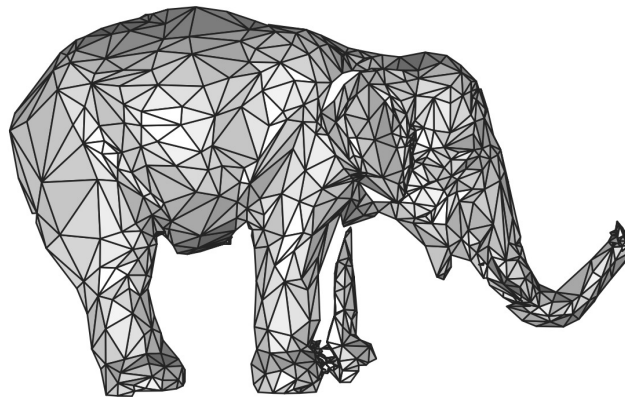


Figure 12.20: Triangular mesh decimation for one rangeview of a real object—a small ceramic sculpture of an elephant. *Courtesy of T. Pajdla, P. Krsek, Czech Technical University, Prague.*

Integration of partial shape descriptions attempts to get known geometric transformations among the views in order to register them and express them in a common co-ordinate system. Precise alignment of the data can be done automatically by gradient minimization, provided good starting transformations are available. In some cases, matching based on invariant features detected on the visual surfaces can be used [Pajdla and Van Gool, 1995], but no method able to cope with a large variety of surfaces has yet been developed.

Figure 12.21 shows approximate manual surface registration, with the help of user interaction. The mutual position of two surfaces is defined by aligning three pairs of matching points; the user selects a few point pairs (the minimum is three) on the surfaces. The approximate registration is obtained by moving one of the surfaces so that the sum of squared distances between the matching point pairs is minimal. An interactive program, *Geomview* (authored by the Geometry Center, University of Minnesota), for 3D surface viewing and manipulation, was used to let the user do the registration. Figures 12.22–12.23 illustrate this procedure.

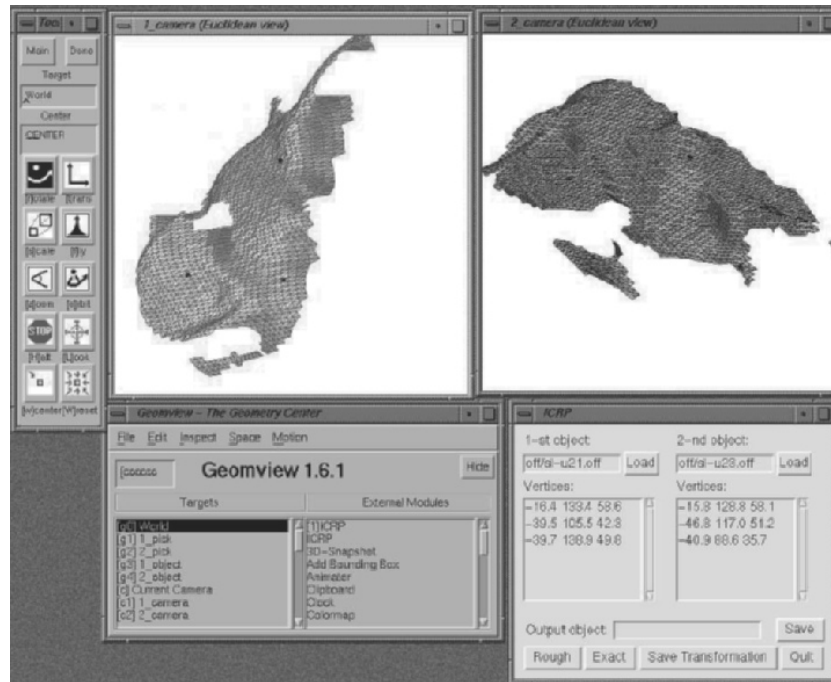


Figure 12.21: Manual registration of surfaces in Geomview. © Cengage Learning 2015.

When two partially overlapping surface patches are roughly registered, automatic refinement of the registration follows. It is assumed that two partially overlapping surfaces P and X related by a global transformation are available. In our case, all transformations are subgroups of a projective group in $\mathcal{P}^3$. Surface registration looks for the best Euclidean transformation T that overlays P and X . T is found by minimization of

$$e = \min_T \rho(P, T(X)) , \quad (12.24)$$

where ρ is a cost function evaluating the quality of match of two surfaces. In Euclidean geometry, it might be the distance between the points on a surface.

The **iterative closest point algorithm** (ICP) [Besl and McKay, 1992] solves the registration automatically provided a good initial estimate of T is available. The algorithm assumes that one of the surfaces is a subset of the second, meaning that only one surface can contain points without correspondence to the second surface. ICP is an iterative optimization procedure that looks for the geometric transformation of one surface to best match the second. It is likely that the cost function will be non-convex, and so there is a consequent danger of falling into local minima—thus a good initial estimate is needed.

We present here a modification of the ICP algorithm which is able to register partial corresponding surfaces. This approach uses the idea of reciprocal points [Pajdla and Van Gool, 1995] to eliminate points without correspondence. Assume point $\mathbf{p}$ is on the surface P and that y is the closest point on the surface X . The closest point on the

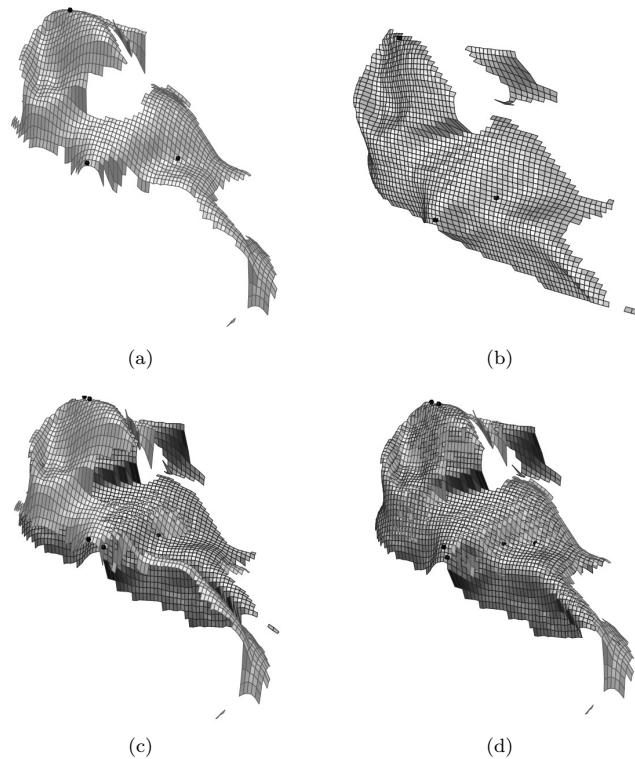


Figure 12.22: The process of registration. (a) Three points on the first surface. (b) Three points on the second surface. (c) Surfaces after rough registration. (d) After exact registration. *Courtesy of T. Pajdla, D. Večerka, Czech Technical University. A color version of this figure may be seen in the color inset—Plate 34.*

surface P to $\mathbf{y}$ is the point $\mathbf{r}$ (see Figure 12.24). Points $\mathbf{p}$, satisfying the condition that the distance is less than ϵ , are called ϵ -reciprocal—only these points are registered. Let P_ϵ denote the set of ϵ -reciprocal points on the surface P ; then the iterative closest reciprocal point algorithm (ICRP) algorithm is as follows.

Algorithm 12.2: Iterative closest reciprocal points

1. Initialize $k = 0$ and $P_0 = P$.
2. Find closest points Y_k for P_k and X .
3. Find reciprocal points $P_{\epsilon 0}$ and $Y_{\epsilon k}$.
4. Compute the mean square distance d_k between $P_{\epsilon k}$ and $Y_{\epsilon k}$.
5. Compute the transformation T between $Y_{\epsilon k}$ and $P_{\epsilon 0}$ in the least-squares sense.

6. Apply the transformation T : $P_{k+1} = T(P_0)$.
7. Compute the mean square distance $d_{k'}$ between P_{k+1} and Y_{ek} .
8. Terminate if the difference $d_k - d_{k'}$ is below a preset threshold or if the maximal number of iterations is exceeded; otherwise go to step 2.

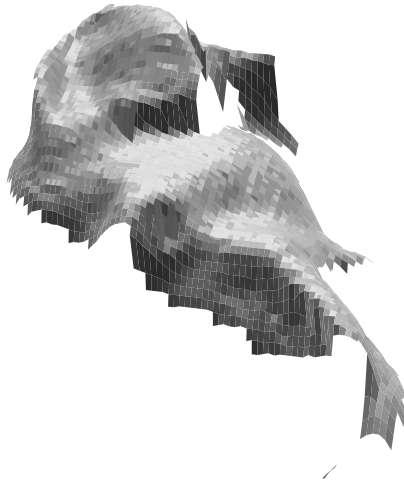


Figure 12.23: The process of registration: rendered result. *Courtesy of T. Pajdla, D. Večerka, Czech Technical University, Prague.*

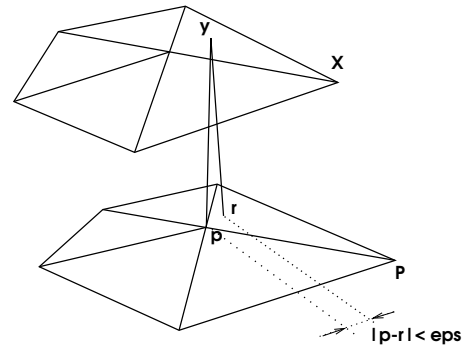


Figure 12.24: The notion of the closest point of the surface for introducing reciprocal points into the ICP algorithm. © Cengage Learning 2015.

When visual surfaces are properly registered, **surface integration** follows. All partial measurements will have been registered and can be expressed in one object-centered co-ordinate system, and constitute a global point. A problem is that the 3D object representation was created from overlapping surface patches corresponding to several views; these patches were integrated following one traversal around the object. All measurements are corrupted by some noise, and this noise is propagated from one surface patch to the other. The important issue is to have approximately the same error when joining the first and last patches. To ensure that the global error is minimal, registered surface points should be rearranged during the surface integration to maintain global consistency.

The next task is to create an analytic shape description of the object's surface by approximation, using some implicit or explicit formulae; commonly, this is not done for the whole object but just for parts of it. A multi-scale local noise model may be used [Šára and Bajcsy, 1998]: an uncertainty ellipsoid (called a fish scale) is used to integrate local information of the point cloud on the surface into a global shape description. The fish scales can be created at multiple resolutions, and their overlap and consistency is explored when creating a 3D shape model.

12.3 2D view-based representations of a 3D scene

12.3.1 Viewing space

Most 3D objects or scenes representations discussed hitherto have been **object-centered**—another option is to use **viewer-centered** representations, where the set of possible appearances of a 3D object is stored as a collection of 2D images. The trouble is that there is potentially an infinite number of possible viewpoints that induce an infinite number of object appearances. To cope with the huge number of viewpoints and appearances it is necessary to sample a viewpoint space and group together similar neighboring views. The original motivation was the recognition of polyhedral objects, which was later generalized to view-based recognition of curved objects [Ullman and Basri, 1991; Beymer and Poggio, 1996].

Consider two models of the viewing space as a representation of possible views on the object or scene. The general model of the viewing space considers all points in a 3D space in which the 3D object is located at the origin. This viewpoint representation is needed if perspective projection is used. A simplified model is a **viewing sphere** model that is often used in the orthographic projection case. Then the object is enclosed by a unit sphere; a point on the sphere's surface gives a viewing direction. The surface can be densely discretized into view patches.

To simplify working with a viewing sphere it is often approximated by a regular polyhedron, of which the most common choice [Horn, 1986] is an icosahedron (with 20 equilateral triangular faces). Twenty viewing directions defined by the centers of the triangles are often not enough, in which case the faces are further regularly divided into four triangles in a recursive manner. This yields 80, 320, 1280, . . . viewing directions.

12.3.2 Multi-view representations and aspect graphs

Other representation methods attempt to combine all the viewpoint-specific models into a single data structure. One of them is the **characteristic view technique** [Chakravarty and Freeman, 1982], in which all possible 2D projections of the convex polyhedral object are grouped into a finite number of topologically equivalent classes. Different views within an equivalence class are related via linear transformations. This general-purpose representation specifies the 3D structure of the object. A similar approach is based on **aspect** [Koenderink and v Doorn, 1979], which is defined as the topological structure of singularities in a single view of an object—aspect has useful invariance properties. Most small changes in vantage point will not affect aspect, and such vantage points (that is, most) are referred to as **stable**. Then the set of stable vantage points yielding a given aspect is contiguous in space, and the space surrounding an object may be partitioned into

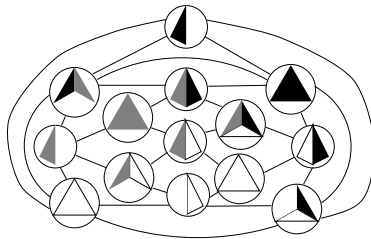


Figure 12.25: Aspect graph for the tetrahedron.
© Cengage Learning 2015.

subspaces from within which the same aspect is generated. An example of the aspect graph for the simplest regular polyhedron—a tetrahedron—is shown in Figure 12.25. Moving from one such subspace into another generates an **event**—a change in aspect; the event is said to connect the two aspects. It is now possible to construct an **aspect graph** with respect to this connectivity relationship in which nodes are aspects and edges are events. This aspect graph is sometimes referred to as the **visual potential** of the object, and paths through it represent orbits of an observer around the object.

12.4 3D reconstruction from an unorganized set of 2D views, and Structure from Motion

A collection of methods is presented in this section which extract the geometric structure of a scene from photographs and provide 3D coordinates of many points on surfaces constituting an (opaque) scene. The ‘images to 3D-points’ data pipeline provides a practical example of the underlying theory of 3D vision geometry introduced in Chapter 11.

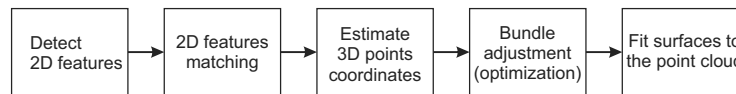


Figure 12.26: Structure from Motion pipeline. © Cengage Learning 2015.

Individual processing steps are arranged into a pipeline of methods/algorithms not requiring any human interaction. The input to the pipeline is an unorganized set of images which overlap in a sufficient number of image pairs, and the output is 3D information about object surfaces. The prerequisite is that there are enough views covering the scene to be reconstructed and that scene objects are sufficiently textured. Camera calibrations and viewer positions are estimated simultaneously with the 3D reconstruction—it is not required to know them in advance. The whole process is passive meaning that no extra energy is transmitted onto the scene. The Structure from Motion (SfM) pipeline is illustrated in Figure 12.26.

An example will illustrate: this uses the Hanau-Kesselstadt lion statue (worked by sculptor Christian Daniel Rauch (1777-1857) belonging to the baroque Palace Philipp-sruhe). The scene was covered by 57 color photos of resolution 2048×1536 pixels captured with an ordinary digital camera—the camera views are illustrated in Figure 12.27. The input images selection of the lion statue is shown in Figure 12.28. The cloud of 3D points in between the views illustrates the outcome of 3D reconstruction (explained in later paragraphs). Each view is represented by one pyramid; its vertex shows the center of projection of a particular camera and the pyramid base shows a particular image captured by this camera.

The (noncommercial) SfM WebService (<http://ptak.felk.cvut.cz/sfmservice>) was used for 3D reconstruction from 57 input images¹. The processing follows the pipeline of Figure 12.26. Interest points and/or interest regions are detected in all input images using the Harris point detector (Section 5.3.10) and/or the Maximally Stable Extremal Regions

¹The SfM WebService was developed by T. Pajdla and his team at the Czech Technical University in Prague. Registered users can submit sequences of images and use the service for 3D reconstruction

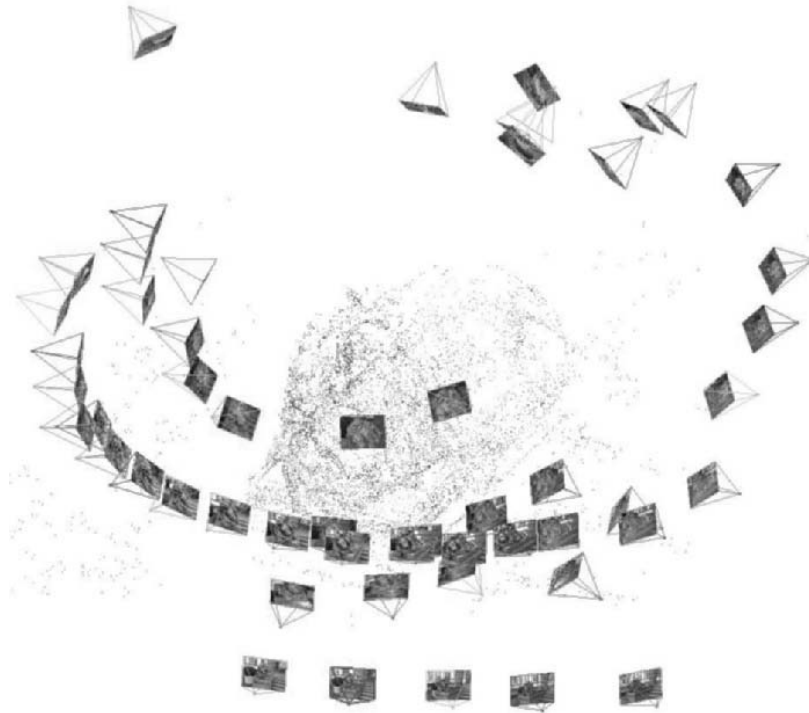


Figure 12.27: Visualization of locations from which the lion scene was photographed. *Courtesy of T. Pajdla, M. Jancosek, C. Albl, Czech Technical University in Prague.*

(MSER) detector (Section 5.3.11). Interest points or MSERs are described using SIFT (Section 10.2).

These descriptors are matched across images exploring the epipolar constraint. A robust RANSAC-based (Section 10.3) procedure is used for this purpose. An example of matches is shown in Figure 12.29.

A set of hundreds of thousands corresponding point pairs candidates serve as input to the algorithm, simultaneously reconstructing 3D points, estimating viewer location and estimating camera calibration parameters. Several different methods are at hand: standard ones [Mičušík and Pajdla, 2006], [Torii et al., 2009a], [Torii et al., 2009b], [Jancosek and Pajdla, 2011] as well as others based on minimal solvers approaching the issue from the algebraic geometry point of view [Kukelova et al., 2012], [Bujnak et al., 2012]. The outcome of 3D reconstruction is a dense 3D point cloud corresponding to detected interest points. Methods are capable of dealing with degenerate situations such as dominant planes, pure camera rotation (panoramas) and zooming. The bundle adjustment method (Section 11.4.4) is used to minimize the reprojection error in many views.

The outcome of the bundle adjustment is a refined cloud of 3D points, which are more consistent after all views are taken into account. The result for the lion statue is illustrated in Figure 12.30. Having the 3D point cloud, the last block of the pipeline



Figure 12.28: Fifteen of fifty seven input images of the lion statue. All fifty seven images are used for reconstruction. *Courtesy of T. Pajdla, M. Jancosek, C. Albl, Czech Technical University in Prague.*



Figure 12.29: Example of correspondences in a pair of images. *Courtesy of T. Pajdla, M. Jancosek, C. Albl, Czech Technical University in Prague.*

from Figure 12.26 comes into play. The 3D points have to be covered by a surface—this is usually done by triangulation covering 3D points by a triangular mesh as explained in Section 12.2.5. Figure 12.31 shows triangulated 3D points from Figure 12.30.

The triangular mesh for a different view on the lion statue is shown in Figure 12.32. The triangular mesh and the information about the color of each triangle in the scene was provided as input to a program permitting viewing of a 3D scene. In this particular case, the program was a VRML (Virtual Reality Modeling Language) viewer. The resulting textured triangular mesh from a view, which is similar to in Figure 12.32, is provided in

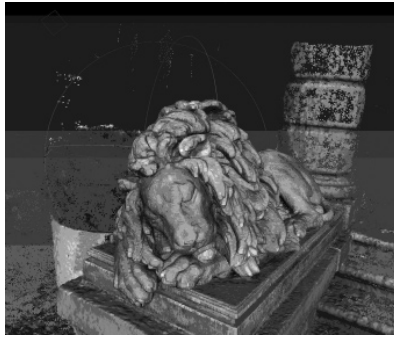


Figure 12.30: One view on the 3D point cloud. Each point is displayed in the color (here gray level) provided by input images. *Courtesy of T. Pajdla, M. Jancosek, C. Albl, Czech Technical University in Prague.*



Figure 12.31: The same view with a triangular mesh built on top of the point cloud from Figure 12.30. *Courtesy of T. Pajdla, M. Jancosek, C. Albl, Czech Technical University in Prague.*



Figure 12.32: Another view on the lion shown as triangular mesh. *Courtesy of T. Pajdla, M. Jancosek, C. Albl, Czech Technical University in Prague.*



Figure 12.33: The same view where the triangular mesh is colored (gray level here) by the appropriate color from input images. *Courtesy of T. Pajdla, M. Jancosek, C. Albl, Czech Technical University in Prague.*

Figure 12.33. The user of the viewer can, however, choose a different view and look at the scene from it.

This section has presented a specific system, but it may be noted that commercial tools such as Street View from Google Inc., or Photosynth by Microsoft Labs and the University of Washington are based on methods explained in this section.

12.5 Reconstructing scene geometry

We have discussed ‘formal’ approaches to 3D reconstruction via matrices characteristic of the camera(s), and homographies: there exist in contrast a range of less formal techniques

which—while lacking the precision we have exercised earlier—serve well in characterizing a scene in view. This is particularly useful in, for example, video sequences of urban scenes, where the objective is an understanding of qualitative behavior of traffic and pedestrians, rather than a precise interpretation of where an agent (some entity moving in the scene, perhaps a vehicle or a pedestrian) might be: to detect congestion or an unexpected event does not require information at the co-ordinate level. This sort of approach becomes more applicable with the prevalence of CCTV, and the widespread application of easily affordable webcams—an approximate knowledge of the groundplane permits an estimation of the speed (and velocity) of objects in motion. We might have discussed this topic in considering motion (Chapter 16), but it is equally well treated here.

A common thread in this approach is the identification of horizon ‘vanishing points’ and then to use established geometric techniques to obtain affine and metric rectification [Hartley and Zisserman, 2003]. Common scenes afford a wide range of opportunities to do this: examples include assuming constant piecewise linear movement [Bose and Grimson, 2003], parallel and orthogonal features of wheelbases [Zhang et al., 2008], assuming bounding boxes surround constant-height humans [Lv et al., 2006; Micusik and Pajdla, 2010], assuming constant road width in traffic scenes [Magee, 2004]. More elaborate approaches build on these approaches, for example by deploying information gain (entropy) arguments to maximize the power of interpretation [Breitenstein et al., 2008].

We will illustrate the idea with a recent application in which common assumptions cannot be exploited. Suppose we have a scene in which geometrical clues, and consistent motion, are unobservable—this is common in congestion where building features are not consistently visible, and individuals regularly occlude each other [Hales et al., 2013]. Street scenes or sports stadia are good examples, in which we would still seek knowledge of the ground geometry and analysis of movement in view.

Some definitions and plausible assumptions are:

- The ground is (effectively) planar. In fact, if it is piecewise planar, an elaboration of this approach would recapture each component.
- The plane is defined in the camera co-ordinate system by the equation $\mathbf{n} \cdot \mathbf{X} = d$ where d is the normal distance of the plane from the origin and $\mathbf{n}$ is the plane normal,

$$\mathbf{n} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sin \psi \sin \theta \\ \cos \psi \sin \theta \\ \cos \theta \end{pmatrix}$$

θ gives the angle of elevation and ψ the yaw.

- Assuming a perspective projection model of the camera (Figure 2.1), the back projection of an image point (x, y) onto the plane is then $\mathbf{X} = (X, Y, Z)$, where

$$\begin{aligned} X &= \alpha x Z \\ Y &= \alpha y Z \\ Z &= \frac{d}{\alpha a x + \alpha b y + c} \end{aligned} \tag{12.25}$$

and α is the negative reciprocal of the camera focal length.

- We are only seeking to reconstruct up to scale, and can set $d = 1$. Further, it is reasonable to assume that the camera is positioned at a reasonable height above the scene.
- Individuals in the scene move at approximately constant speed. Different individuals may have different speeds, but we may expect them to be comparable.
- Individuals can be tracked over limited time periods—perhaps a small number of seconds—before they are lost by occlusion. These short image trajectories $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ will be assumed to be linear segments of a 3D trajectory on the ground plane, sampled at regular time intervals.

The image segment $\mathbf{x}_{t-1}, \mathbf{x}_t$ then represents a distance L_t in 3D, where from equation (12.25)

$$L_t^2 = \alpha^2 \left(\frac{x_t}{\gamma_t} - \frac{x_{t-1}}{\gamma_{t-1}} \right)^2 + \alpha^2 \left(\frac{y_t}{\gamma_t} - \frac{y_{t-1}}{\gamma_{t-1}} \right)^2 + \left(\frac{1}{\gamma_t} - \frac{1}{\gamma_{t-1}} \right)^2, \quad (12.26)$$

where

$$\gamma_t = \alpha x_t a + \alpha y_t b + c.$$

A particular trajectory T defines a set of such distances L^T : if our assumptions are correct, all elements of L^T are equal—thus the standard deviation of this set $\sigma(L^T)$ is of interest. Choices of α, θ, ψ that minimize this quantity may be regarded as a good match to the data. In fact, we wish to choose parameters to minimize this measure over all trajectories: since different objects may move at different speeds, we seek to minimize

$$E_1 = \sum_T \left(\frac{\sigma(L^T)}{\mu(L^T)} \right)^2 \quad (12.27)$$

—that is, normalizing the standard deviation by the mean length $\mu(L_T)$. Since we expect agents to move with comparable speed, we also seek to minimize the variance of means

$$E_2 = \sigma(\mu(L_T)). \quad (12.28)$$

Recovery of the ground plane parameters thus becomes a minimization of $E_1 + \lambda E_2$ for some λ . It is not surprising that the problem space is complex, and common non-linear minimization algorithms fall into poor solutions. On the other hand, a coarse-to-fine strategy of quantizing (α, θ, ψ) space generates much better solutions: the original reference has details [Hales et al., 2013].

In implementation, the popular KLT tracker (see Section 16.3.2) is used to deliver image trajectories: very short ones are discarded, and clusters of similar trajectories (perhaps coming from the same individual, or from agents moving collectively) are clustered. Empirically, it was discovered that setting the error-weight parameter to 1 delivered acceptable results. When benchmarked against well-known test-sets [CVPR, 2009], errors in θ and ψ rarely exceed 10° and are usually much less. Relative speeds deduced from solution parameters show greater plausibility than those recovered from other popular algorithms [Bose and Grimson, 2003]. Figure 12.34 illustrates one example.

This example shows how a sufficient quantity of vague information, with suitable reasoning, can deliver real-world information of value in the absence of formal calibration. Algorithms of this kind permit cheap consumer electronics to be deployed easily and to discover enough about what they are surveying to deliver useful 3D information. A very good and early grounding in this technique may be found in [Criminisi, 1999].

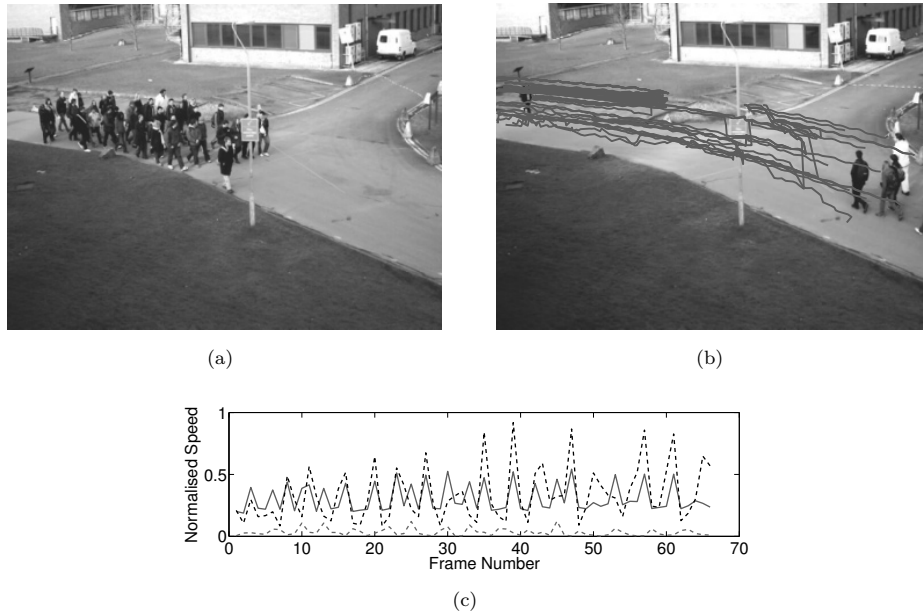


Figure 12.34: Use of scene geometry for people movement tracking. (a) A still from one of the PETS sequences [CVPR, 2009]. (b) From the same sequence—some KLT tracks overlaid. (c) Normalized speeds—ground truth in black and this algorithm in red (solid). For comparison, the green line is derived from [Bose and Grimson, 2003]. *Courtesy of I. Hales, University of Leeds. A color version of this figure may be seen in the color inset—Plate 35.*

12.6 Summary

- We have seen an overview of this topic: We have given a taxonomy of various 3D vision tasks, a summary of current approaches, formulated tasks, and shown some applications.
- **Shape from X**
 - Shape may be extracted from motion, optical flow, texture, focus/de-focus, vergence, and contour.
 - Each of these techniques may be used to derive a 2.5D sketch for Marr's vision theory; they are also of practical use on their own.
- **Full 3D objects**
 - Line labeling is an outmoded but accessible technique for reconstructing objects with planar faces.
 - Transitions to 3D objects need a co-ordinate system that is object centered.
 - 3D objects may be measured mechanically by computed tomography, by range-finders or by shape from motion techniques.

- Volumetric modeling strategies include constructive solid geometry, superquadrics and generalized cylinders.
- Surface modeling strategies include boundary representations, triangulated surfaces, and quadric patches.
- **3D model-based vision**
 - To create a full 3D model from a set of range images, the surfaces must first be registered—rotations and translations should be found that match one surface to another.
 - Model-based vision uses a priori knowledge about an object to ease its recognition.
 - Techniques exist to locate curved objects from range images.
- **2D view-based representations of a 3D scene**
 - 2D view-based representations of 3D scenes may be achieved with multi-view representations.
 - It is possible to select a few stored reference images, and render any view from them.
 - Interpolation of views is not enough and view extrapolation is needed. This requires knowledge of geometry, and the view-based approach does not differ significantly from 3D geometry reconstruction.
 - It is possible to perform a 3D reconstruction from an unorganized set of 2D views. This approach has been used widely recently by, e.g., Google Street View.
- **Reconstructing scene geometry**
 - Large scale scene features such as plane parameters may be recaptured from properties of known objects such as straight lines and approximate size.
 - Well known geometric results identify vanishing points and ground orientation.
 - Similar approaches may well work even if large scale clues are unavailable.

12.7 Exercises

Short-answer questions

- S12.1 For each of the following *shape from X* techniques, briefly describe the principle underlying it, explain how they can be of practical use, and give some examples.
- Motion
 - Texture
 - Focus
 - Vergence
 - Contour

- S12.2 Give a counter-example in which apparent motion measured by an optical flow field does not correspond to actual motion of the 3D points. [Hint: Consider an opaque sphere, where (1) the sphere does not move and illumination changes; (2) the sphere rotates and the illumination is constant.]
- S12.3 Fashion designers cheat the human ability to derive shape from texture. How do they do it?
- S12.4 Give an example of a rim which is not a planar curve.
- S12.5 Are there line drawings that do not correspond to physically plausible polyhedral objects? If yes, show an example.
- S12.6 What was the contribution of image interpretation via line labeling to computer vision? Where else can it be used and why? (Hint: model-based 3D object recognition)
- S12.7 Describe how to create a full 3D model of an object from range images taken from different views.
- S12.8 Outline how a reconstruction of a 3D scene from unorganized set of 2D views may be acquired. What steps are needed?

Problems

- P12.1 Conduct a laboratory experiment that establishes depth from focus. Take a fixed-focus lens with small depth of focus, and switch off the auto-focus feature of the lens if possible. Open the aperture as much as possible; mount the camera with the lens on a stand that can adjust the distance between the camera and the object.
- Select a small window in the image that corresponds to a point in the scene from which the distance is measured. Devise an algorithm that assists in achieving the sharpest image. (Hint: The sharpest image has the most high frequencies; is there some method to find such an image that is simpler than the Fourier transform?)
- Finally, the distance to the scene surface corresponding to the patch can be read from the mechanical scale on the stand.
- P12.2 Most computer vision techniques measure co-ordinates of distinct points on a 3D object surface, e.g., those where correspondences were found; the result is a 3D point cloud. The human notion of shape is not of a point cloud. Elaborate on this problem, e.g., in the case of human face where 3D data are measured using stereopsis. How would you move from a point cloud to a surface? Is the surface the shape we seek? (Hint: Shape is treated from the geometer's point of view in [Koenderink, 1990]).
- P12.3 Using a standard digital camera, take some shots of a 3D object of your choice and exercise the software at (<http://ptak.felk.cvut.cz/sfmservice>).
- P12.4 Make yourself familiar with solved problems and **Matlab** implementations of selected algorithms provided in the corresponding chapter of the **Matlab Companion** to this text [Svoboda et al., 2008]. The **Matlab Companion** homepage <http://visionbook.felk.cvut.cz> offers images used in the problems, and well-commented **Matlab** code is provided for educational purposes.
- P12.5 Use the **Matlab Companion** [Svoboda et al., 2008] to develop solutions to additional exercises and practical problems provided there. Implement your solutions using **Matlab** or other suitable programming languages.

12.8 References

- Aloimonos J. and Swain M. J. Shape from texture. In *9th International Joint Conference on Artificial Intelligence*, Los Angeles, volume 2, pages 926–931, Los Altos, CA, 1985. Morgan Kaufmann Publishers.
- Aloimonos Y., editor. *Active Perception*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.
- Bajcsy R. and Lieberman L. Texture gradient as a depth cue. *Computer Graphics and Image Processing*, 5:52–67, 1976.
- Barr A. H. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.
- Besl P. J. and Jain R. C. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, 1985.
- Besl P. J. and McKay N. D. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.
- Beymer D. and Poggio T. Image representations for visual learning. *Science*, 272:1905–1909, 1996.
- Binford T. O. Visual perception by computer. In *Proceedings of the IEEE Conference on Systems, Science and Cybernetics*, Miami, FL, 1971. IEEE.
- Blake A. and Yuille A. *Active Vision*. MIT Press, Cambridge, MA, 1992.
- Blostein D. and Ahuja N. Shape from texture: Integrating texture-element extraction and surface estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1233–1251, 1989.
- Bose B. and Grimson E. Ground plane rectification by tracking moving objects. In *IEEE International Workshop on Visual Surveillance and PETS*, 2003.
- Bowyer K. W. Special issue on directions in CAD-based vision. *CVGIP – Image Understanding*, 55:107–218, 1992.
- Breitenstein M. D., Sommerlade E., Leibe B., Gool L. J. V., and Reid I. Probabilistic parameter selection for learning scene structure from video. In Everingham M., Needham C. J., and Fraile R., editors, *Proceedings of the British Machine Vision Conference 2008, Leeds, September 2008*, pages 1–10. British Machine Vision Association, 2008.
- Brooks R. A., Greiner R., and Binford T. O. The ACRONYM model-based vision system. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-6*, Tokyo, Japan, pages 105–113, 1979.
- Bujnak M., Kukulova Z., and Pajdla T. Efficient solutions to the absolute pose of cameras with unknown focal length and radial distortion by decomposition to planar and non-planar cases. *IPSP Transactions on Computer Vision and Applications*, 4:78–86, May 2012. ISSN 1882-6695.
- Chakravarty I. and Freeman H. Characteristic views as a basis for three-dimensional object recognition. *Proceedings of The Society for Photo-Optical Instrumentation Engineers Conference on Robot Vision*, 336:37–45, 1982.
- Clocksink W. F. Perception of surface slant and edge labels from optical flow—a computational approach. *Perception*, 9:253–269, 1980.
- Clowes M. B. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.
- Criminisi A. *Accurate Visual Metrology from Single and Multiple Uncalibrated Images*. Ph.D. thesis, University of Oxford, Dept. Engineering Science, 1999. D.Phil. thesis.
- CVPR. PETS 2009 data set, 2009. <http://www.cvg.rdg.ac.uk/PETS2009/a.html>.

- Faugeras O. D. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA, 1993.
- Flynn P. J. and Jain A. K. 3D object recognition using invariant feature indexing of interpretation tables. *CVGIP – Image Understanding*, 55(2):119–129, 1992.
- Gubais L. J., Knuth D. E., and M S. Randomized incremental construction of Delunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.
- Hales I., Hogg D., Ng K. C., and Boyle R. D. Automated ground-plane estimation for trajectory rectification. In *Proceedings of CAIP 2013, York UK*, volume II. Springer Verlag, LNCS 8048, 2013.
- Hartley R. I. and Zisserman A. *Multiple view geometry in computer vision*. Cambridge University, Cambridge, 2nd edition, 2003.
- Higuchi K., Hebert M., and Ikeuchi K. Building 3-D models from unregistered range images. *Graphics Models and Image Processing*, 57(4):313–333, 1995.
- Hoppe H., DeRose T., Duchamp T., McDonald J., and Stuetzle W. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.
- Horn B. K. P. *Robot Vision*. MIT Press, Cambridge, MA, 1986.
- Huffman D. A. Impossible objects as nonsense sentences. In Metzler B. and Michie D. M., editors, *Machine Intelligence*, volume 6, pages 295–323. Edinburgh University Press, Edinburgh, 1971.
- Jaklič A. *Construction of CAD models from Range Images*. Ph.D. thesis, Department of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia, 1997.
- Jancosek M. and Pajdla T. Multi-view reconstruction preserving weakly-supported surfaces. In Felzenszwalb P., Forsyth D., and Fua P., editors, *CVPR 2011: Proceedings of the 2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3121–3128, New York, USA, June 2011. IEEE Computer Society, IEEE Computer Society. ISBN 978-1-4577-0393-5. doi: 10.1109/CVPR.2011.599569.
- Kanatani K. and Chou T. C. Shape from texture: General principle. *Artificial Intelligence*, 38(1):1–48, 1989.
- Klir G. J. *Facets of System Science*. Plenum Press, New York, 1991.
- Koenderink J. J. *Solid Shape*. MIT Press, Cambridge, MA, 1990.
- Koenderink J. J. and Doorn A. J. v. Internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32(4):211–216, 1979.
- Kriegman D. J. Computing stable poses of piecewise smooth objects. *CVGIP – Image Understanding*, 55(2):109–118, 1992.
- Krotkov E. P. Focusing. *International Journal of Computer Vision*, 1:223–237, 1987.
- Krotkov E. P. and Bajcsy R. Active vision for reliable ranging: Cooperating focus, stereo, and vergence. *International Journal of Computer Vision*, 11(2):187–203, 1993.
- Kukelova Z., Bujnak M., and Pajdla T. Polynomial eigenvalue solutions to minimal problems in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1381–1393, July 2012. ISSN 0162-8828.
- Leonardis A., Jaklič A., and Solina F. Superquadrics for segmenting and modeling range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1289–1295, 1997.
- Lv F., Zhao T., and Nevatia R. Camera calibration from video of a walking human. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9), 2006.

- Magee D. R. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 22(2):143–155, 2004.
- Malik J. and Maydan D. Recovering three-dimensional shape from a single image of curved object. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):555–566, 1989.
- Marr D. *Vision—A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman, San Francisco, 1982.
- Mičušík B. and Pajdla T. Structure from motion with wide circular field of view cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1135–1149, July 2006. ISSN 0162-8828.
- Micusik B. and Pajdla T. Simultaneous surveillance camera calibration and foot-head homology estimation from human detections. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1562–1569, Los Alamitos, CA, USA, 2010. IEEE Computer Society.
- Nayar S. K. and Nakagawa Y. Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):824–831, 1994.
- Nayar S. K., Watanabe M., and Hoguchi M. Real-time focus range sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1186–1197, 1996.
- Negahdaripour S. and Horn B. K. P. Determining 3D motion of planar objects from image brightness measurements. In *9th International Joint Conference on Artificial Intelligence*, Los Angeles, volume 2, pages 898–901, Los Altos, CA, 1985. Morgan Kaufmann Publishers.
- Nevatia R., Zerroug M., and Ulupinar F. Recovery of three-dimensional shape of curved objects from a single image. In Young T. Y., editor, *Handbook of Pattern Recognition and Image Processing: Computer Vision*, pages 101–129, San Diego, CA, 1994. Academic Press.
- Pajdla T. and Van Gool L. Matching of 3-D curves using semi-differential invariants. In *5th International Conference on Computer Vision*, Boston, USA, pages 390–395. IEEE Computer Society, 1995.
- Pentland A. A new sense for depth of field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):523–531, 1987.
- Ponce J., Hoogs A., and Kriegman D. J. On using CAD models to compute the pose of curved 3d objects. *CVGIP – Image Understanding*, 55(2):184–197, 1992.
- Preparata F. P. and Shamos M. I. *Computational Geometry—An Introduction*. Springer Verlag, Berlin, 1985.
- Rissanen J. *Stochastic Complexity in Statistical Inquiry*. World Scientific, Series in Computer Science, IBM Almaden Research Center, San Jose, CA, 1989.
- Roberts L. G. Machine perception of three-dimensional solids. In Tippet J. T., editor, *Optical and Electro-Optical Information Processing*, pages 159–197. MIT Press, Cambridge, MA, 1965.
- Šára R. and Bajcsy R. Fish-scales: Representing fuzzy manifolds. In Chandran S. and Desai U., editors, *Proceedings of the 6th International Conference on Computer Vision (ICCV)*, pages 811–817, New Delhi, India, January 1998. IEEE Computer Society, Narosa Publishing House.
- Sato K., Ikeuchi K., and Kanade T. Model based recognition of specular objects using sensor models. *CVGIP – Image Understanding*, 55(2):155–169, 1992.
- Seales W. B. and Dyer C. R. Viewpoints from occluding contour. *CVGIP – Image Understanding*, 55(2):198–211, 1992.

- Shomar W. J. and Young T. Y. Three-dimensional shape recovery from line drawings. In *Handbook of Pattern Recognition and Image Processing: Computer Vision*, volume 2, pages 53–100, San Diego, CA, 1994. Academic Press.
- Solina F. and Bajcsy R. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–147, 1990.
- Soroka B. I. and Bajcsy R. K. A program for describing complex three dimensional objects using generalised cylinders. In *Proceedings of the Pattern Recognition and Image Processing Conference* Chicago, pages 331–339, New York, 1978. IEEE.
- Soucy M. and Laurendeau D. Multiresolution surface modeling based on hierarchical triangulation. *Computer Vision and Image Understanding*, 63(1):1–14, 1996.
- Stevens K. A. Representing and analyzing surface orientation. In Winston P. A. and Brown R. H., editors, *Artificial Intelligence: An MIT Perspective*, volume 2. MIT Press, Cambridge, MA, 1979.
- Sugihara K. *Machine Interpretation of Line Drawings*. MIT Press, Cambridge, MA, 1986.
- Svoboda T., Kybic J., and Hlavac V. *Image Processing, Analysis, and Machine Vision: A MATLAB Companion*. Thomson Engineering, 2008.
- Terzopoulos D. and Fleischer K. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.
- Terzopoulos D. and Metaxas D. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991.
- Terzopoulos D., Witkin A., and Kass M. Constraints on deformable models: Recovering 3-D shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.
- Torii A., Havlena M., and Pajdla T. From google street view to 3d city models. In *OMNIVIS '09: 9th IEEE Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, page 8, Los Alamitos, USA, October 2009a. IEEE Computer Society Press. ISBN 978-1-4244-4441-0. CD-ROM.
- Torii A., Havlena M., and Pajdla T. Omnidirectional image stabilization by computing camera trajectory. In Wada T., Huang F., and Lin S. Y., editors, *PSIVT '09: Advances in Image and Video Technology: Third Pacific Rim Symposium*, volume 5414 of *Lecture Notes in Computer Science*, pages 71–82, Berlin, Germany, January 2009b. Springer Verlag. ISBN 978-3-540-92956-7.
- Ullman S. *The Interpretation of Visual Motion*. MIT Press, Cambridge, MA, 1979.
- Ullman S. *High-Level Vision: Object Recognition and Visual Cognition*. MIT Press, Cambridge, MA, 1996.
- Ullman S. and Basri R. Recognition by linear combination of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1005, 1991.
- Uray P. *From 3D point clouds to surface and volumes*. Ph.D. thesis, Technische Universitaet Graz, Austria, 1997.
- Waltz D. L. Understanding line drawings of scenes with shadows. In Winston P. H., editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, New York, 1975.
- Witkin A. P. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17: 17–45, 1981.
- Zhang Z., Li M., Huang K., and Tan T. Robust automated ground plane rectification based on moving vehicles for traffic scene surveillance. In *Proceedings of the International Conference on Image Processing, ICIP 2008, October 12-15, 2008, San Diego, California, USA*, pages 1364–1367, 2008.

Chapter 13

Mathematical morphology

13.1 Basic morphological concepts

Mathematical morphology is a well-established area of image-analysis that stands separately from the mainstream. It is based on the algebra of non-linear operators operating on object shape and in many respects supersedes the linear algebraic system of convolution. It performs in many tasks—pre-processing, segmentation using object shape, and object quantification—better and more quickly than the standard approaches. There is a rich background literature: [Serra, 1982, 1987; Giardina and Dougherty, 1988; Dougherty, 1992; Heijmans, 1994].

Our approach is introductory [Haralick and Shapiro, 1992; Vincent, 1995; Soille, 2003]. Morphological tools are implemented in most image analysis packages, and we give the reader enough to apply them in a qualified way. Mathematical morphology is very often used in applications where shape of objects and speed is an issue—for example, analysis of microscopic images (in biology, material science, geology, and criminology), industrial inspection, optical character recognition, and document analysis.

The non-morphological approach to image processing is calculus-related, being based on the point-spread function concept and linear transformations such as convolution, and we have discussed image modeling and processing from this point of view in other chapters. Mathematical morphology uses tools of non-linear algebra and operates with point sets, their connectivity and shape. Morphological operations simplify images, and quantify and preserve the main shape characteristics of objects.

Morphological operations are used predominantly for the following purposes:

- Image pre-processing (noise filtering, shape simplification).
- Enhancing object structure (skeletonizing, thinning, thickening, convex hull,).
- Segmenting objects from the background.
- Quantitative description of objects (area, perimeter, projections,).

Mathematical morphology exploits point set properties, results of integral geometry, and topology. The initial assumption states that real images can be modeled using **point sets** of any dimension (e.g., N -dimensional Euclidean space); the Euclidean 2D space $\mathcal{E}^2$ and its system of subsets is a natural domain for planar shape description. Standard set algebra is used; **set difference** is defined by

$$X \setminus Y = X \cap Y^c. \quad (13.1)$$

Computer vision uses the digital counterpart of Euclidean space—sets of integer pairs ($\in \mathcal{Z}^2$) for binary image morphology or sets of integer triples ($\in \mathcal{Z}^3$) for gray-scale morphology or binary 3D morphology.

We begin by considering binary images that can be viewed as subsets of the 2D space of all integers, $\mathcal{Z}^2$. A point is represented by a pair of integers that give co-ordinates with respect to the two co-ordinate axes of the digital raster; the unit length of the raster equals the sampling period in each direction. We talk about a *discrete grid* if the neighborhood relation between points is well defined. This representation is suitable for both rectangular and hexagonal grids, but a rectangular grid is assumed hereafter.

A binary image can be treated as a 2D point set. Points belonging to objects in the image represent a set X —these points are pixels with value equal to one. Points of the complement set X^c correspond to the background with pixel values equal to zero. The origin (marked as a diagonal cross in our examples) has co-ordinates $(0, 0)$, and co-ordinates of any point are interpreted as (x, y) in the common way used in mathematics. Figure 13.1 shows an example of such a set—points belonging to the object are denoted by small black squares. Any point x from a discrete image $X = \{(1, 0), (1, 1), (1, 2), (2, 2), (0, 3), (0, 4)\}$ can be treated as a vector with respect to the origin $(0, 0)$.

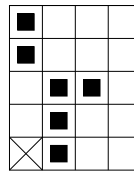


Figure 13.1: A point set example. © Cengage Learning 2015.

A **morphological transformation** Ψ is given by the relation of the image (point set X) with another small point set B called a **structuring element**. B is expressed with respect to a local origin $\mathcal{O}$ (called the representative point). Some typical structuring elements are shown in Figure 13.2. Figure 13.2c illustrates the possibility of the point $\mathcal{O}$ not being a member of the structuring element B .

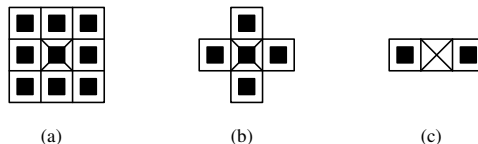


Figure 13.2: Typical structuring elements. © Cengage Learning 2015.

To apply the morphological transformation $\Psi(X)$ to the image X means that the structuring element B is moved systematically across the entire image. Assume that B is positioned at some point in the image; the pixel in the image corresponding to the representative point $\mathcal{O}$ of the structuring element is called the **current pixel**. The result

of the relation (which can be either zero or one) between X and B in the current position is stored in the output image in the current image pixel position.

The **duality** of morphological operations is deduced from the existence of the set complement; for each morphological transformation $\Psi(X)$ there exists a dual transformation $\Psi^*(X)$:

$$\Psi(X) = (\Psi^*(X^c))^c. \quad (13.2)$$

The **translation** of the point set X by the vector h is denoted by X_h ; it is defined by

$$X_h = \{p \in \mathcal{E}^2, p = x + h \text{ for some } x \in X\}. \quad (13.3)$$

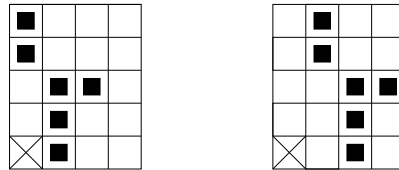


Figure 13.3: Translation by a vector.
© Cengage Learning 2015.

13.2 Four morphological principles

It is appropriate to restrict the set of possible morphological transformations in image analysis by imposing several constraints; we shall present here four morphological principles that express such constraints. A full understanding of these is not essential to a comprehension of what follows, and they may be taken for granted. A detailed explanation of these matters may be found in [Serra, 1982].

Humans have an intuitive understanding of *spatial structure*—the structure of the Alps versus an oak tree crown is perceived as different. Besides the need for objective descriptions of such objects, the scientist requires a quantitative description. Generalization is expected as well; the interest is not in a specific oak tree, but in the class of oaks.

The morphological approach with quantified results consists of two main steps: (a) geometrical transformation and (b) the actual measurement. [Serra, 1982] gives two examples. The first is from chemistry, where the task is to measure the surface area of some object. First, the initial body is reduced to its surface, e.g., by marking by some chemical matter. Second, the quantity of the marker needed to cover the surface is measured. Another example is from sieving analysis, often used in geology when the distribution of sizes of milled rocks is of interest. The milled product is passed through sieves with different sizes of holes from coarse to fine. The result is a sequence of subsets of milled product. For each sieve size some oversize particles remain on it after sieving, and these are measured.

A morphological operator is (by definition) composition of a mapping Ψ (or geometrical transformation) followed by a measure μ which is a mapping $Z \times \dots \times Z \rightarrow R$. The geometrically transformed set $\Psi(X)$ can be the boundary, oversized particles in sieving analysis, etc., and the measure $\mu[\Psi(X)]$ yields a number (weight, surface area, volume, etc.). The discussion here is simplified just to the transformation Ψ , but the axiomatics can be transposed to measures as well.

A morphological transformation is called **quantitative** if and only if it satisfies four basic principles [Serra, 1982].

- **Compatibility with translation:** Let the transformation Ψ depend on the position of the origin $\mathcal{O}$ of the co-ordinate system, and denote such a transformation by $\Psi_{\mathcal{O}}$. If all points are translated by the vector $-h$, it is expressed as Ψ_{-h} . The **compatibility with translation** principle is given by

$$\Psi_{\mathcal{O}}(X_h) = (\Psi_{-h}(X))_h . \quad (13.4)$$

If Ψ does not depend on the position of the origin $\mathcal{O}$, then the compatibility with translation: principle reduces to invariance under translation

$$\Psi(X_h) = (\Psi(X))_h . \quad (13.5)$$

- **Compatibility with change of scale:** Let λX represent the scaling of a point set X . This is equivalent to change of scale with respect to some origin. Let Ψ_{λ} denote a transformation that depends on the positive parameter λ (change of scale). **Compatibility with change of scale** is given by

$$\Psi_{\lambda}(X) = \lambda \Psi\left(\frac{1}{\lambda} X\right) . \quad (13.6)$$

If Ψ does not depend on the scale λ , then compatibility with change of scale reduces to invariance to change of scale

$$\Psi(\lambda X) = \lambda \Psi(X) . \quad (13.7)$$

- **Local knowledge:** The local knowledge principle considers the situation in which only a part of a larger structure can be examined—this is always the case in reality, due to the restricted size of the digital grid. The morphological transformation Ψ satisfies the **local knowledge principle** if for any bounded point set Z' in the transformation $\Psi(X)$ there exists a bounded set Z , knowledge of which is sufficient to provide Ψ . The local knowledge principle may be written symbolically as

$$(\Psi(X \cap Z)) \cap Z' = \Psi(X) \cap Z' . \quad (13.8)$$

- **Upper semi-continuity:** The upper semi-continuity principle says that the morphological transformation does not exhibit any abrupt changes. A precise explanation needs many concepts from topology and is given in [Serra, 1982].

13.3 Binary dilation and erosion

Sets of black and white pixels constitute a binary image. Assume that only black pixels are considered, and the others are treated as a background. The primary morphological operations are dilation and erosion, and from these two, more complex morphological operations such as opening, closing, and shape decomposition can be derived. We present them here using Minkowski's formalism [Haralick and Shapiro, 1992]. The Minkowski algebra is closer to the notions taught in standard mathematics courses (an alternative is Serra's formalism based on stereological concepts [Serra, 1982]).

13.3.1 Dilation

The morphological transformation **dilation** $\oplus$ combines two sets using vector addition (or Minkowski set addition, e.g., $(a, b) + (c, d) = (a + c, b + d)$). The dilation $X \oplus B$ is the point set of all possible vector additions of pairs of elements, one from each of the sets X and B

$$X \oplus B = \{p \in \mathcal{E}^2 : p = x + b, x \in X \text{ and } b \in B\}. \quad (13.9)$$

Figure 13.4 illustrates an example of dilation:

$$\begin{aligned} X &= \{(1, 0), (1, 1), (1, 2), (2, 2), (0, 3), (0, 4)\}, \\ B &= \{(0, 0), (1, 0)\}, \\ X \oplus B &= \{(1, 0), (1, 1), (1, 2), (2, 2), (0, 3), (0, 4), (2, 0), (2, 1), (2, 2), (3, 2), (1, 3), (1, 4)\}. \end{aligned}$$

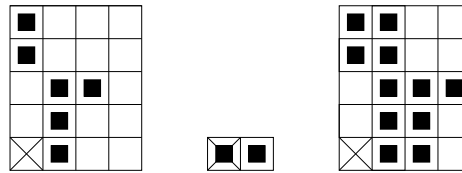


Figure 13.4: Dilation. © Cengage Learning 2015.

Figure 13.5 shows a 256×256 original image (the emblem of the Czech Technical University) on the left. A structuring element of size 3×3 , see Figure 13.2a, is used. The result of dilation is shown on the right side of Figure 13.5. In this case the dilation is an *isotropic* expansion (it behaves the same way in all directions). This operation is also sometimes called *fill* or *grow*.

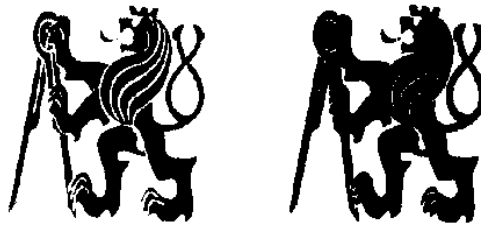


Figure 13.5: Dilation as isotropic expansion. © Cengage Learning 2015.

Dilation with an isotropic 3×3 structuring element might be described as a transformation which changes all background pixels neighboring the object to object pixels.

Dilation has several interesting properties that may ease its hardware or software implementation; we present some here without proof. The interested reader may consult [Serra, 1982] or the tutorial paper [Haralick et al., 1987].

The dilation operation is commutative

$$X \oplus B = B \oplus X \quad (13.10)$$

and is also associative

$$X \oplus (B \oplus D) = (X \oplus B) \oplus D. \quad (13.11)$$

Dilation may also be expressed as a union of shifted point sets

$$X \oplus B = \bigcup_{b \in B} X_b \quad (13.12)$$

and is invariant to translation

$$X_h \oplus B = (X \oplus B)_h. \quad (13.13)$$

Equations (13.12) and (13.13) show the importance of shifts in speeding up implementation of dilation, and this holds for implementations of binary morphology on serial computers in general. One processor word represents several pixels (e.g., 32 for a 32-bit processor), and shift or addition corresponds to a single instruction. Shifts may also be easily implemented as delays in a pipeline parallel processor.

Dilation is an **increasing** transformation:

$$\text{If } X \subseteq Y \text{ then } X \oplus B \subseteq Y \oplus B. \quad (13.14)$$

Dilation is used to fill small holes and narrow gulfs in objects. It increases the object size—if the original size needs to be preserved, then dilation is combined with erosion, described in the next section.

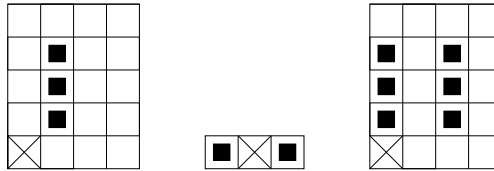


Figure 13.6: Dilation where the representative point is not a member of the structuring element. © Cengage Learning 2015.

Figure 13.6 illustrates the result of dilation if the representative point is not a member of the structuring element B ; if this structuring element is used, the dilation result is substantially different from the input set. Notice that the connectivity of the original set has been lost.

13.3.2 Erosion

Erosion $\ominus$ combines two sets using vector subtraction of set elements and is the dual operator of dilation. Neither erosion nor dilation is an invertible transformation

$$X \ominus B = \{p \in \mathcal{E}^2 : p + b \in X \text{ for every } b \in B\}. \quad (13.15)$$

This formula says that every point p from the image is tested; the result of the erosion is given by those points p for which all possible $p + b$ are in X . Figure 13.7 shows an example of the point set X eroded by the structuring element B :

$$\begin{aligned} X &= \{(1, 0), (1, 1), (1, 2), (0, 3), (1, 3), (2, 3), (3, 3), (1, 4)\}, \\ B &= \{(0, 0), (1, 0)\}, \\ X \ominus B &= \{(0, 3), (1, 3), (2, 3)\}. \end{aligned}$$

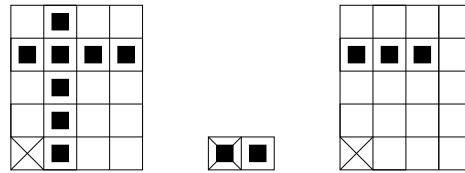


Figure 13.7: Erosion.
© Cengage Learning 2015.



Figure 13.8: Erosion as isotropic shrink. © Cengage Learning 2015.

Figure 13.8 shows erosion by a 3×3 element (see Figure 13.2a) of the same original as Figure 13.5. Notice that single-pixel-wide lines disappear. Erosion with an isotropic structuring element is called **shrink** or **reduce** by some authors.

Basic morphological transformations can be used to find the contours of objects in an image very quickly. This can be achieved, for instance, by subtraction from the original picture of its eroded version—see Figure 13.9.



Figure 13.9: Contours obtained by subtraction of an eroded image from an original (left). © Cengage Learning 2015.

Erosion is used to simplify the structure of an object—objects or their parts with width equal to one will disappear. It might thus decompose complicated objects into several simpler ones.

There is an equivalent definition of erosion [Matheron, 1975]. Recall that B_p denotes B translated by p

$$X \ominus B = \{p \in \mathcal{E}^2 : B_p \subseteq X\}. \quad (13.16)$$

The erosion might be interpreted by structuring element B sliding across the image X ; then, if B translated by the vector p is contained in the image X , the point corresponding to the representative point of B belongs to the erosion $X \ominus B$.

An implementation of erosion might be simplified by noting that an image X eroded by the structuring element B can be expressed as an intersection of all translations of

the image X by the vector¹ $-b \in B$

$$X \ominus B = \bigcap_{b \in B} X_{-b} . \quad (13.17)$$

If the representative point is a member of the structuring element, then erosion is an anti-extensive transformation; that is, if $(0,0) \in B$, then $X \ominus B \subseteq X$. Erosion is also translation invariant

$$X_h \ominus B = (X \ominus B)_h , \quad (13.18)$$

$$X \ominus B_h = (X \ominus B)_{-h} , \quad (13.19)$$

and, like dilation, is an increasing transformation:

$$\text{If } X \subseteq Y \text{ then } X \ominus B \subseteq Y \ominus B . \quad (13.20)$$

If B , D are structuring elements, and D is contained in B , then erosion by B is more aggressive than by D ; that is, if $D \subseteq B$, then $X \ominus B \subseteq X \ominus D$. This property enables the ordering of erosions according to structuring elements of similar shape but different sizes.

Denote by $\check{B}$ the *symmetrical set* to B (also called the *transpose* [Serra, 1982] or *rational set* [Haralick et al., 1987]) with respect to the representative point $\mathcal{O}$

$$\check{B} = \{-b : b \in B\} . \quad (13.21)$$

For example

$$\begin{aligned} B &= \{(1,2), (2,3)\} , \\ \check{B} &= \{(-1,-2), (-2,-3)\} . \end{aligned} \quad (13.22)$$

We have already mentioned that erosion and dilation are dual transformations. Formally

$$(X \ominus Y)^C = X^C \oplus \check{Y} . \quad (13.23)$$

The differences between erosion and dilation are illustrated by the following properties. Erosion (in contrast to dilation) is not commutative

$$X \ominus B \neq B \ominus X . \quad (13.24)$$

The properties of erosion and intersection combined together are

$$\begin{aligned} (X \cap Y) \ominus B &= (X \ominus B) \cap (Y \ominus B) , \\ B \ominus (X \cap Y) &\supseteq (B \ominus X) \cup (B \ominus Y) . \end{aligned} \quad (13.25)$$

On the other hand, image intersection and dilation cannot be interchanged; the dilation of the intersection of two images is contained in the intersection of their dilations

$$(X \cap Y) \oplus B = B \oplus (X \cap Y) \subseteq (X \oplus B) \cap (Y \oplus B) . \quad (13.26)$$

¹This definition of erosion, $\ominus$, differs from that used in [Serra, 1982]. There $\ominus$ denotes Minkowski subtraction, which is an intersection of all translations of the image by the vector $b \in B$. In our case the minus sign has been added. In our notation, if convex sets are used, the dilation of erosion (or the other way around) is identity.

The order of erosion may be interchanged with set union. This fact enables the structuring element to be decomposed into a union of simpler structuring elements

$$\begin{aligned} B \oplus (X \cup Y) &= (X \cup Y) \oplus B = (X \oplus B) \cup (Y \oplus B), \\ (X \cup Y) \ominus B &\supseteq (X \ominus B) \cup (Y \ominus B), \\ B \ominus (X \cup Y) &= (X \ominus B) \cap (Y \ominus B). \end{aligned} \quad (13.27)$$

Successive dilation (respectively, erosion) of the image X first by the structuring element B and then by the structuring element D is equivalent to the dilation (erosion) of the image X by $B \oplus D$:

$$\begin{aligned} (X \oplus B) \oplus D &= X \oplus (B \oplus D), \\ (X \ominus B) \ominus D &= X \ominus (B \oplus D). \end{aligned} \quad (13.28)$$

13.3.3 Hit-or-miss transformation

The hit-or-miss transformation is a morphological operator for finding local patterns of pixels, where *local* means the size of the structuring element. It is a variant of template matching that finds collections of pixels with certain shape properties (such as corners, or border points), and can also be used for thinning and thickening of objects (Section 13.5.3).

Operations described hitherto used a structuring element B , and we have tested points for their membership of X ; we can also test whether some points do not belong to X . An operation may be denoted by a pair of disjoint sets $B = (B_1, B_2)$, called a **composite structuring element**. The **hit-or-miss** transformation $\otimes$ is defined as

$$X \otimes B = \{x : B_1 \subset X \text{ and } B_2 \subset X^c\}. \quad (13.29)$$

This means that for a point x to be in the resulting set, two conditions must be fulfilled simultaneously. First the part B_1 of the composite structuring element that has its representative point at x must be contained in X , and second, the part B_2 of the composite structuring element must be contained in X^c .

The hit-or-miss transformation operates as a binary matching between an image X and the structuring element (B_1, B_2) . It may be expressed using erosions and dilations as well

$$X \otimes B = (X \ominus B_1) \cap (X^c \ominus B_2) = (X \ominus B_1) \setminus (X \oplus \check{B}_2). \quad (13.30)$$

13.3.4 Opening and closing

Erosion and dilation are not inverse transformations—if an image is eroded and then dilated, the original image is not re-obtained. Instead, the result is a simplified and less detailed version of the original image.

Erosion followed by dilation creates an important morphological transformation called **opening**. The opening of an image X by the structuring element B is denoted by $X \circ B$ and is defined as

$$X \circ B = (X \ominus B) \oplus B. \quad (13.31)$$

Dilation followed by erosion is called **closing**. The closing of an image X by the structuring element B is denoted by $X \bullet B$ and is defined as

$$X \bullet B = (X \oplus B) \ominus B. \quad (13.32)$$

If an image X is unchanged by opening with the structuring element B , it is called *open with respect to B* . Similarly, if an image X is unchanged by closing with B , it is called *closed with respect to B* .

Opening and closing with an isotropic structuring element is used to eliminate specific image details smaller than the structuring element—the global shape of the objects is not distorted. Closing connects objects that are close to each other, fills up small holes, and smooths the object outline by filling up narrow gulfs. Meanings of ‘near’, ‘small’, and ‘narrow’ are related to the size and the shape of the structuring element. Opening is illustrated in Figure 13.10, and closing in Figure 13.11.



Figure 13.10: Opening (original on the left). © Cengage Learning 2015.



Figure 13.11: Closing (original on the left). © Cengage Learning 2015.

Unlike dilation and erosion, opening and closing are invariant to translation of the structuring element. Equations (13.14) and (13.20) imply that both opening and closing are increasing transformations. Opening is anti-extensive ($X \circ B \subseteq X$) and closing is extensive ($X \subseteq X \bullet B$).

Opening and closing, like dilation and erosion, are dual transformations

$$(X \bullet B)^C = X^C \circ \check{B}. \quad (13.33)$$

Iteratively used openings and closings are **idempotent**, meaning that reapplication of these transformations does not change the previous result. Formally:

$$X \circ B = (X \circ B) \circ B, \quad (13.34)$$

$$X \bullet B = (X \bullet B) \bullet B. \quad (13.35)$$

13.4 Gray-scale dilation and erosion

Binary morphological operations acting on binary images are easily extendible to gray-scale images using the ‘min’ and ‘max’ operations. Erosion (respectively, dilation) of an image is the operation of assigning to each pixel the minimum (maximum) value found over a neighborhood of the corresponding pixel in the input image. The structuring element is more rich than in the binary case, where it gave only the neighborhood. In the gray-scale case, the structuring element is a function of two variables that specifies the desired local gray-level property. The value of the structuring element is added (subtracted) when the maximum (or minimum) is calculated in the neighborhood.

This extension permits a *topographic view* of gray-scale images—the gray-level is interpreted as the height of a particular location of a hypothetical landscape: light and dark spots in the image correspond to landscape hills and hollows. This morphological approach permits the location of global image properties, i.e., to identify characteristic topographic features on images as valleys, mountain ridges (crests), and watersheds.

13.4.1 Top surface, umbra, and gray-scale dilation and erosion

Consider a point set A in n -dimensional Euclidean space, $A \subset \mathcal{E}^n$, and assume that the first $(n - 1)$ co-ordinates of the set constitute a spatial domain and the n^{th} co-ordinate corresponds to the value of a function or functions at a point ($n = 3$ for gray-scale images). This interpretation matches the topographic view for a 3D Euclidean space, where points are given by triples of co-ordinates; the first two co-ordinates locate the position in the 2D support set and the third co-ordinate gives the height.

The **top surface** of a set A is a function defined on the $(n - 1)$ -dimensional support. For each $(n - 1)$ -tuple, the top surface is the highest value of the last co-ordinate of A , as illustrated in Figure 13.12. If the space is Euclidean the highest value means supremum.

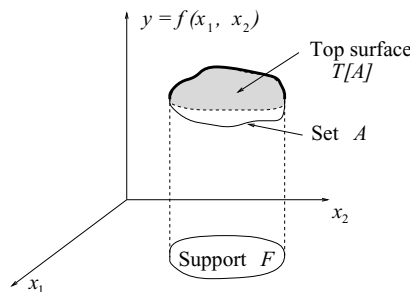


Figure 13.12: Top surface of the set A corresponds to maximal values of the function $f(x_1, x_2)$. © Cengage Learning 2015.

Let $A \subseteq \mathcal{E}^n$ and the support $F = \{x \in \mathcal{E}^{n-1} \text{ for some } y \in \mathcal{E}, (x, y) \in A\}$. The *top surface* of A , denoted by $T[A]$, is a mapping $F \rightarrow \mathcal{E}$ defined as

$$T[A](x) = \max \{y, (x, y) \in A\}. \quad (13.36)$$

An **umbra** is a region of complete shadow resulting from obstructing light by a non-transparent object. In mathematical morphology, the umbra of a function f is a set that consists of the top surface of f and everything below it; see Figure 13.13.

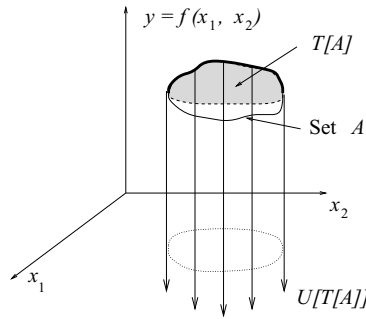


Figure 13.13: Umbra of the top surface of a set is the whole subspace below it. © Cengage Learning 2015.

Formally, let $F \subseteq \mathcal{E}^{n-1}$ and $f : F \rightarrow \mathcal{E}$. The *umbra* of f , denoted by $U[f]$, $U[f] \subseteq F \times \mathcal{E}$, is defined by

$$U[f] = \{(x, y) \in F \times \mathcal{E}, y \leq f(x)\}. \quad (13.37)$$

We see that the umbra of an umbra of f is an umbra.

We can illustrate the top surface and umbra in the case of a simple 1D gray-scale image. Figure 13.14 illustrates a function f (which might be a top surface) and its umbra.

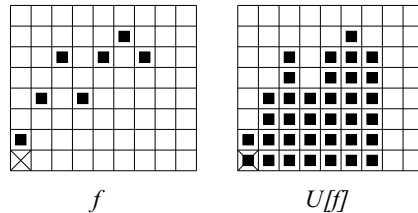


Figure 13.14: Example of a 1D function (left) and its umbra (right). © Cengage Learning 2015.

We can now define the gray-scale dilation of two functions as the top surface of the dilation of their umbras. Let $F, K \subseteq \mathcal{E}^{n-1}$ and $f : F \rightarrow \mathcal{E}$ and $k : K \rightarrow \mathcal{E}$. The *dilation* $\oplus$ of f by k , $f \oplus k : F \oplus K \rightarrow \mathcal{E}$ is defined by

$$f \oplus k = T\{U[f] \oplus U[k]\}. \quad (13.38)$$

Notice here that $\oplus$ on the left-hand side is dilation in the *gray-scale* image domain, and $\oplus$ on the right-hand side is dilation in the *binary* image.

Similarly to binary dilation, one function, say f , represents an image, and the second, k , a small structuring element. Figure 13.15 shows a discretized function k that will play the role of the structuring element. Figure 13.16 shows the dilation of the umbra of f (from the example given in Figure 13.14) by the umbra of k .

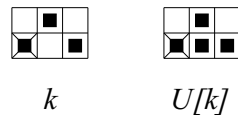


Figure 13.15: A structuring element: 1D function (left) and its umbra (right). © Cengage Learning 2015.

This definition explains what gray-scale dilation means, but does not give a reasonable algorithm for actual computations. We shall see that a computationally plausible

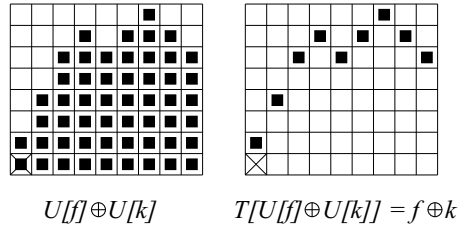


Figure 13.16: 1D example of gray-scale dilation. The umbras of the 1D function f and structuring element k are dilated first, $U[f] \oplus U[k]$. The top surface of this dilated set gives the result, $f \oplus k = T[U[f] \oplus U[k]]$.
© Cengage Learning 2015.

way to calculate dilation can be obtained by taking the maximum of a set of sums:

$$(f \oplus k)(x) = \max \{f(x - z) + k(z), z \in K, x - z \in F\}. \quad (13.39)$$

The computational complexity is the same as for convolution in linear filtering, where a summation of products is performed.

The definition of **gray-scale erosion** is analogous to dilation. The gray-scale erosion of two functions (point sets):

1. Takes their umbras.
2. Erodes them using binary erosion.
3. Gives the result as the top surface.

Let $F, K \subseteq \mathcal{E}^{n-1}$ and $f : F \rightarrow \mathcal{E}$ and $k : K \rightarrow \mathcal{E}$. The *erosion* $\ominus$ of f by k , $f \ominus k : F \ominus K \rightarrow \mathcal{E}$ is defined by

$$f \ominus k = T\{U[f] \ominus U[k]\}. \quad (13.40)$$

Erosion is illustrated in Figure 13.17. To decrease computational complexity, the actual computations are performed in another way as the minimum of a set of differences (notice the similarity to correlation):

$$(f \ominus k)(x) = \min_{z \in K} \{f(x + z) - k(z)\}. \quad (13.41)$$

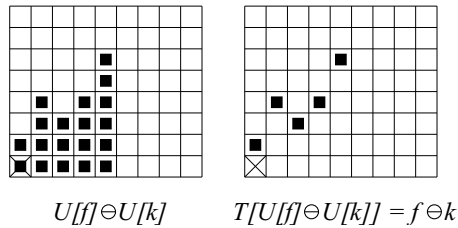


Figure 13.17: 1D example of gray-scale erosion. The umbras of 1D function f and structuring element k are eroded first, $U[f] \ominus U[k]$. The top surface of this eroded set gives the result, $f \ominus k = T[U[f] \ominus U[k]]$.
© Cengage Learning 2015.

We illustrate morphological pre-processing on a microscopic image of cells corrupted by noise in Figure 13.18a; the aim is to reduce noise and locate individual cells. Figure 13.18b shows erosion of the original image, and Figure 13.18c illustrates dilation of the original image. A 3×3 structuring element was used in both cases—notice that the noise has been considerably reduced. The individual cells can be located by the reconstruction operation (to be explained in Section 13.5.4). The original image is used as a mask and the dilated image in Figure 13.18c is an input for reconstruction. The result is shown in image 13.18d, in which the black spots depict the cells.

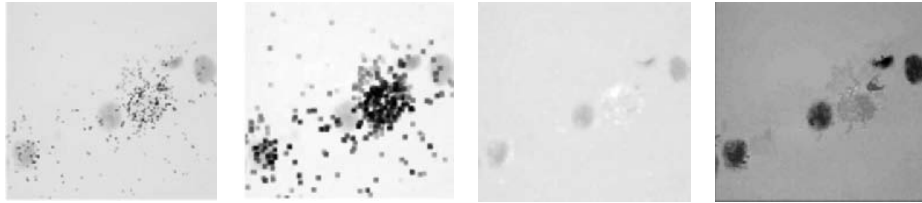


Figure 13.18: Morphological pre-processing: (a) cells in a microscopic image corrupted by noise; (b) erosion of original image, the noise is more prominent; (c) dilation of original image, the noise has disappeared; (d) reconstructed cells. *Courtesy of P. Kodl, Rockwell Automation Research Center, Prague.*

13.4.2 Umbra homeomorphism theorem, properties of erosion and dilation, opening and closing

The top surface always inverts the umbra operation; i.e., the top surface is a left inverse of the umbra, $T[U[f]] = f$. However, the umbra is not an inverse of the top surface. The strongest conclusion that can be deduced is that the umbra of the top surface of a point set A contains A (recall Figure 13.13).

The notion of top surface and umbra provides an intuitive relation between gray-scale and binary morphology. The **umbra homeomorphism theorem** states that the umbra operation is a homeomorphism from gray-scale morphology to binary morphology. Let $F, K \subseteq \mathcal{E}^{n-1}$, $f : F \rightarrow \mathcal{E}$, and $k : K \rightarrow \mathcal{E}$. Then [Haralick and Shapiro, 1992]:

$$\begin{aligned} \text{(a)} \quad U[f \oplus k] &= U[f] \oplus U[k], \\ \text{(b)} \quad U[f \ominus k] &= U[f] \ominus U[k]. \end{aligned} \quad (13.42)$$

The umbra homeomorphism is used for deriving properties of gray-scale operations. The operation is expressed in terms of umbra and top surface, then transformed to binary sets using the umbra homeomorphism property, and finally transformed back using the definitions of gray-scale dilation and erosion. Using this idea, properties already known from binary morphology can be derived, e.g., commutativity of dilation, the chain rule that permits decomposition of large structural elements into successive operations with smaller ones, duality between erosion and dilation.

Gray-scale opening and closing is defined in the same way as in the binary morphology. **Gray-scale opening** is defined as $f \circ k = (f \ominus k) \oplus k$. Similarly, **gray-scale closing** $f \bullet k = (f \oplus k) \ominus k$. The *duality* between opening and closing is expressed as (recall that $\check{k}$ means the transpose, i.e., symmetric set with regards to origin of co-ordinates)

$$-(f \circ k)(x) = ((-f) \bullet \check{k})(x). \quad (13.43)$$

There is a simple geometric interpretation of gray-scale opening [Haralick and Shapiro, 1992]. The opening of f by structuring element k can be interpreted as sliding k on the landscape f . The position of all highest points reached by some part of k during the slide gives the opening, and a similar interpretation exists for erosion.

Gray-scale opening and closing is often used in applications to extract parts of a gray-scale image with given shape and gray-scale structure.

13.4.3 Top hat transformation

The top hat transformation is used as a simple tool for segmenting objects in gray-scale images that differ in brightness from background, even when the background is of uneven gray-scale. The top hat transform is superseded by the watershed segmentation (to be described in Section 13.7.3) for more complicated backgrounds.

Assume a gray-level image X and a structuring element K . The residue of opening as compared to the original image $X \setminus (X \circ K)$ constitutes an operation called a *top hat transformation* [Meyer, 1978].

The top hat transformation is a good tool for extracting light objects (or, conversely, dark ones) on a dark (or light) but slowly changing background. Those parts of the image that cannot fit into structuring element K are removed by opening. Subtracting the opened image from the original provides an image where removed objects stand out clearly. The actual segmentation can be performed by simple thresholding. The concept is illustrated for the 1D case in Figure 13.19, where we can see the origin of the transformation name. If an image were a hat, the transformation would extract only the top of it, provided that the structuring element is larger than the hole in the hat.

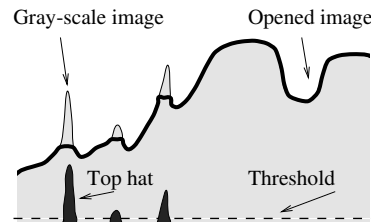


Figure 13.19: The top hat transform permits the extraction of light objects from an uneven background. © Cengage Learning 2015.

Figure 13.20 provides an example from visual industrial inspection. A factory producing glass capillaries for mercury maximal thermometers had the following problem: The thin glass tube should be narrowed in one particular place to prevent mercury falling back when the temperature decreases from the maximal value. This is done by using

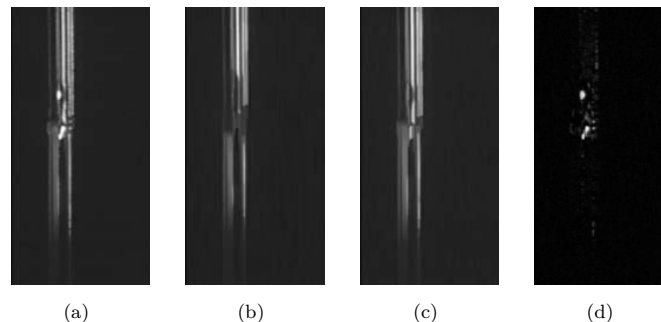


Figure 13.20: An industrial example of gray-scale opening and top hat segmentation, i.e., image-based control of glass tube narrowing by gas flame. (a) Original image of the glass tube, 512×256 pixels. (b) Erosion by a one-pixel-wide vertical structuring element 20 pixels long. (c) Opening with the same element. (d) Final specular reflection segmentation by the top hat transformation. Courtesy of V. Smutný, R. Šára, CTU Prague, P. Kodl, Rockwell Automation Research Center, Prague.

a narrow gas flame and low pressure in the capillary. The capillary is illuminated by a collimated light beam—when the capillary wall collapses due to heat and low pressure, an instant specular reflection is observed and serves as a trigger to cover the gas flame. Originally the machine was controlled by a human operator who looked at the tube image projected optically on the screen; the gas flame was covered when the specular reflection was observed. This task had to be automated and the trigger signal learned from a digitized image. The specular reflection is detected by a morphological procedure.

13.5 Skeletons and object marking

13.5.1 Homotopic transformations

Topological properties are associated with continuity (Section 2.3.1), and mathematical morphology can be used to study such properties of objects in images. There is an interesting group among morphological transformations called **homotopic transformations** [Serra, 1982].

A transformation is homotopic if it does not change the continuity relation between regions and holes in the image. This relation is expressed by the homotopic tree; its root corresponds to the background of the image, first-level branches correspond to the objects (regions), second-level branches match holes within objects, etc. Figure 13.21

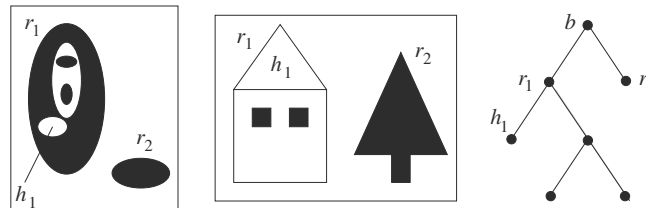


Figure 13.21: The same homotopic tree for two different images. © Cengage Learning 2015.

shows an example of two images having the same homotopic tree. On the left side are some biological cells and at center a house and a spruce tree. The tree's root b corresponds to the background, node r_1 matches the larger cell (the outline of the house), and node r_2 matches the smaller cell (the spruce tree). Node h_1 corresponds to the empty hole in the cell r_1 (the hole inside the roof of the house)—the other correspondences to nodes should now be clear. A transformation is homotopic if it does not change the homotopic tree.

13.5.2 Skeleton, medial axis, maximal ball

It is sometimes advantageous to convert an object to an archetypal stick figure called a **skeleton** (also considered in Section 8.3.4). We shall explain this in the context of 2D Euclidean space first, which is more illustrative than on the digital grid that we shall consider later.

The skeleton was introduced under the name **medial axis transform** [Blum, 1967] and illustrated on a 'grassfire' scenario: Assume a region (point set) $X \subset \mathcal{R}^2$: a grassfire is lit on the entire region boundary at the same instant, and propagates towards the

region interior with constant speed. The skeleton $S(X)$ is the set of points where two or more firefronts meet; see Figure 13.22.

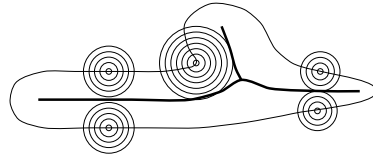


Figure 13.22: Skeleton as points where two or more firefronts of grassfire meet. © Cengage Learning 2015.

A more formal definition of skeleton is based on the concept of maximal ball. A **ball** $B(p, r)$ with center p and radius r , $r \geq 0$, is the set of points with distances d from the center less than or equal to r . The ball B included in a set X is said to be *maximal* if and only if there is no larger ball included in X that contains B , i.e., each ball B' , $B \subseteq B' \subseteq X \implies B' = B$. Balls and maximal balls are illustrated in Figure 13.23.

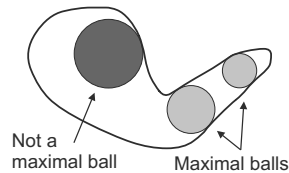


Figure 13.23: Ball and two maximal balls in a Euclidean plane. © Cengage Learning 2015.

The distance metric d that is used depends on the grid and definition of connectivity. Unit balls in a plane (i.e., unit disks) are shown in Figure 13.24.

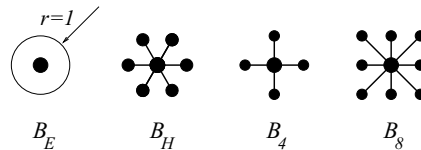


Figure 13.24: Unit-size disk for different distances, from left side: Euclidean distance, 6-, 4-, and 8-connectivity, respectively. © Cengage Learning 2015.

The plane $\mathcal{R}^2$ with the usual Euclidean distance gives the ball B_E . Three distances and balls are often defined in the discrete plane $\mathcal{Z}^2$. If a hexagonal grid and 6-connectivity is used, the hexagonal ball B_H is obtained. If the support is a square grid, two unit balls are possible: B_4 for 4-connectivity and B_8 for 8-connectivity.

The *skeleton by maximal balls* $S(X)$ of a set $X \subset \mathcal{Z}^2$ is the set of centers p of maximal balls

$$S(X) = \{p \in X : \exists r \geq 0, B(p, r) \text{ is a maximal ball of } X\}.$$

This definition of skeleton has an intuitive meaning in the Euclidean plane. The skeleton of a disk reduces to its center, the skeleton of a stripe with rounded endings is a unit thickness line at its center, etc.

Figure 13.25 shows several objects together with their skeletons—a rectangle, two touching balls, and a ring. The properties of the (Euclidean) skeleton can be seen here—in particular, the skeleton of two adjacent circles consists of three distinct points instead of a straight line joining these two points, as might be intuitively expected.

The skeleton by maximal balls has two unfortunate properties in practical applications. First, it does not necessarily preserve the homotopy (connectivity) of the original

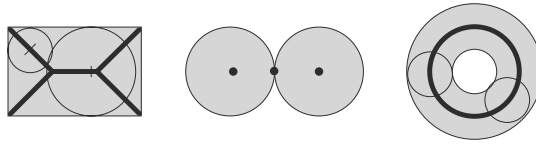


Figure 13.25: Skeletons of rectangle, two touching balls, and a ring.
© Cengage Learning 2015.

set; and second, some of the skeleton lines may be wider than one pixel in the discrete plane. We shall see later that the skeleton is often substituted by sequential homotopic thinning that does not have these two properties.

Dilation can be used in any of the three discrete connectivities to create balls of varying radii. Let nB be the ball of radius n

$$nB = B \oplus B \oplus \dots \oplus B. \quad (13.44)$$

The skeleton by maximal balls can be obtained as the union of the residues of opening of the set X at all scales [Serra, 1982]

$$S(X) = \bigcup_{n=0}^{\infty} \left((X \ominus nB) \setminus (X \ominus nB) \circ B \right). \quad (13.45)$$

The trouble with this is that the resulting skeleton is completely disconnected and this property is not useful in many applications. Thus *homotopic skeletons* that preserve connectivity are often preferred (Section 13.5.3).

13.5.3 Thinning, thickening, and homotopic skeleton

One application of the hit-or-miss transformation (Section 13.3.3) is **thinning** and **thickening** of point sets. For an image X and a composite structuring element $B = (B_1, B_2)$ (notice that B here is not a ball), *thinning* is defined as

$$X \oslash B = X \setminus (X \otimes B) \quad (13.46)$$

and *thickening* is defined as

$$X \odot B = X \cup (X^c \otimes B). \quad (13.47)$$

When thinning, a part of the boundary of the object is subtracted from it by the set difference operation. When thickening, a part of the boundary of the background is added to the object. Thinning and thickening are dual transformations

$$(X \odot B)^c = X^c \oslash B, \quad B = (B_2, B_1). \quad (13.48)$$

Thinning and thickening transformations are very often used sequentially. Let $\{B_{(1)}, B_{(2)}, B_{(3)}, \dots, B_{(n)}\}$ denote a sequence of composite structuring elements $B_{(i)} = (B_{i1}, B_{i2})$. *Sequential thinning* can then be expressed as a sequence of n structuring elements for square rasters

$$X \oslash \{B_{(i)}\} = \left(((X \oslash B_{(1)}) \oslash B_{(2)}) \dots \oslash B_{(n)} \right) \quad (13.49)$$

and *sequential thickening* as

$$X \odot \{B_{(i)}\} = \left(((X \odot B_{(1)}) \odot B_{(2)}) \dots \odot B_{(n)} \right). \quad (13.50)$$

There are several sequences of structuring elements $\{B_{(i)}\}$ that are useful in practice. Many of them are given by a permissible rotation of a structuring element in the appropriate digital raster (e.g., hexagonal, square, or octagonal). These sequences, sometimes called the **Golay alphabet** [Golay, 1969], are summarized for the hexagonal raster in [Serra, 1982]. We shall present 3×3 matrices for the first two rotations of the Golay alphabet for octagonal rasters, from which the other rotations can easily be derived.

A composite structuring element can be expressed by a single matrix only. A value of one in it means that this element belongs to B_1 (it is a subset of objects in the hit-or-miss transformation), and a value zero belongs to B_2 and is a subset of the background. An asterisk $*$ in the matrix denotes an element that is not used in the matching process, so its value is not significant.

Thinning and thickening sequential transformations converge to some image—the number of iterations needed depends on the objects in the image and the structuring element used. If two successive images in the sequence are identical, the thinning (or thickening) is stopped.

Sequential thinning by structuring element L

This thinning is quite important, as it serves as the homotopic substitute of the skeleton; the final image consists only of lines of width one and isolated points.

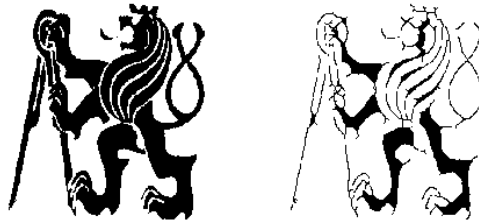


Figure 13.26: Sequential thinning using element L after five iterations. © Cengage Learning 2015.

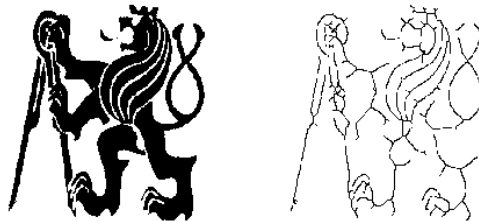


Figure 13.27: Homotopic substitute of the skeleton (element L). © Cengage Learning 2015.

The structuring element L from the Golay alphabet is given by

$$L_1 = \begin{bmatrix} 0 & 0 & 0 \\ * & 1 & * \\ 1 & 1 & 1 \end{bmatrix}, \quad L_2 = \begin{bmatrix} * & 0 & 0 \\ 1 & 1 & 0 \\ * & 1 & * \end{bmatrix}, \quad \dots \quad (13.51)$$

(The other six elements are given by rotation.) Figure 13.26 shows the result of five iterations of thinning with L , and Figure 13.27 shows the homotopic substitute of the skeleton when idempotency was reached (in both cases, the original is shown on the left).

Sequential thinning by structuring element E

Assume that the homotopic substitute of the skeleton by element L has been found. The skeleton is usually jagged, because of sharp points on the outline of the object, but it is possible to ‘smooth’ it by sequential thinning by structuring element E . Using n iterations, several points (whose number depends on n) are removed from the ends of lines of width one (and isolated points as well). If thinning by element E is performed until the image does not change, then only closed contours remain.

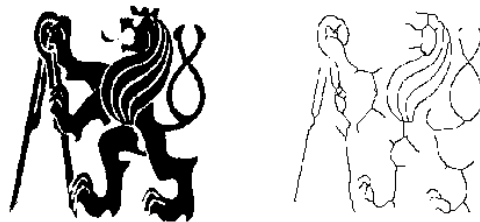


Figure 13.28: Five iterations of sequential thinning by element E .
© Cengage Learning 2015.

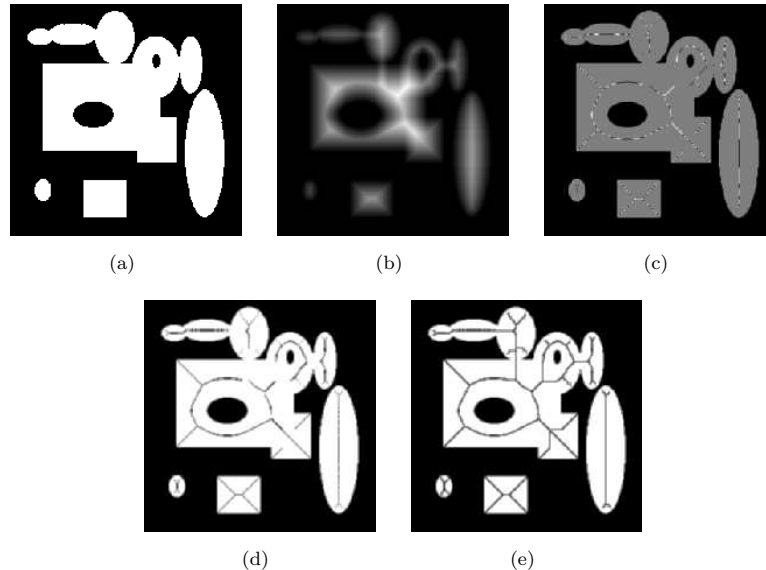


Figure 13.29: Performance of Vincent's quick skeleton by maximal balls algorithm. (a) Original binary image. (b) Distance function (to be explained later). (c) Distance function visualized by contouring. (d) Non-continuous skeleton by maximal balls. (e) Final skeleton. © Cengage Learning 2015.

The structuring element E from the Golay alphabet is given again by rotated masks,

$$E_1 = \begin{bmatrix} * & 1 & * \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad E_2 = \begin{bmatrix} 0 & * & * \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \dots \quad (13.52)$$

Figure 13.28 shows sequential thinning (five iterations) by the element E of the skeleton from Figure 13.27. Notice that lines have been shortened from their free ends.

There are three other elements M , D , C in the Golay alphabet [Golay, 1969] which are not much used in practice. Other morphological algorithms are used instead to find skeletons, convex hulls, and homotopic markers.

The computationally most efficient algorithm of which we are aware creates the connected skeleton as the minimal superset of the skeleton by maximal balls [Vincent, 1991]. Its performance is shown in Figure 13.29: homotopy is preserved.

The skeleton can be applied to native 3D images as well, e.g., in the analysis of computer tomography images. Figure 13.30 illustrates examples of thinning of 3D point sets; parallel algorithms are available [Ma and Sonka, 1996; Palagyi et al., 2006].

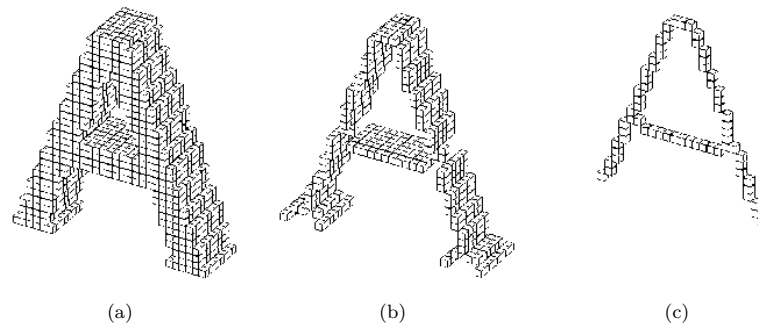


Figure 13.30: Morphological thinning in 3D. (a) Original 3D data set, a character A. (b) Thinning performed in one direction. (c) One voxel thick skeleton obtained by thinning image (b) in second direction. *Courtesy of K. Palagyi, University of Szeged, Hungary.*

13.5.4 Quench function, ultimate erosion

The binary point set X can be described using maximal balls B . Every point p of the skeleton $S(X)$ by maximal balls has an associated ball of radius $q_X(p)$; the term **quench function** is used for this association. An important property of $q_X(p)$ is that the quench function permits the reconstruction of the original set X completely as a union of its maximal balls B

$$X = \bigcup_{p \in S(X)} (p + q_X(p)B). \quad (13.53)$$

This formula allows lossless compression of a binary image. Similar ideas are used for encoding documents in CCITT group 4 compression algorithms.

It is useful to distinguish several types of extrema, and use the analogy of a topographic view of images to make the explanation more intuitive. The *global maximum* is

the pixel with highest value (lightest pixel, highest summit in the countryside); similarly, the *global minimum* corresponds to the deepest chasm in the countryside.

A pixel p of a gray-scale image is a *local maximum* if and only if for every neighboring pixel q of a pixel p , $I(p) \geq I(q)$. For example, the local maximum may mean that the landscape around is studied in a small neighborhood of the current position (neighborhood in morphology is defined by the structuring element). If no ascent is seen within the neighborhood, the pixel is at a local maximum.

The *regional maximum* M of a digital gray-scale image I is a connected set of pixels with an associated value h (plateau at altitude h), such that every neighboring pixel of M has strictly lower value than h . Topographically, regional extrema correspond to geographic summits and hollows. If M is a regional maximum of I and $p \in M$, then p is a local maximum, but the converse does not hold. Local and regional maxima are illustrated for the 1D case in Figure 13.31. The definition of various maxima allows us to analyze the quench function, which is also useful to define **ultimate erosion**, which is often used as a marker of convex objects in binary images. The ultimate erosion of a set X , denoted $\text{Ult}(X)$, is the set of regional maxima of the quench function. The natural markers are centers of the largest maximal balls, but a problem arises if the objects are overlapping—here ultimate erosion comes into play. Consider first the simplest case, in which the set X consists of two overlapping disks (see Figure 13.32). The skeleton is a line segment between the centers. The associated quench function has regional maxima that are located at the disk centers in this particular example. These maxima are called ultimate erosion and can be used as markers of overlapping objects. Ultimate erosion provides a tool that extracts one marker per object of a given shape, even if objects overlap. The remaining trouble is that some objects are still multiply marked.

Consider a binary image, a set X , consisting of three rounded overlapping components of different size. When iteratively eroding the set by a unit-size ball, the set is shrunk, then separates, and finally disappears, as illustrated in Figure 13.33. During the successive erosions, the residuals of connected components (just before they disappear) are stored. Their union is the ultimate erosion of the original set X (Figure 13.34).

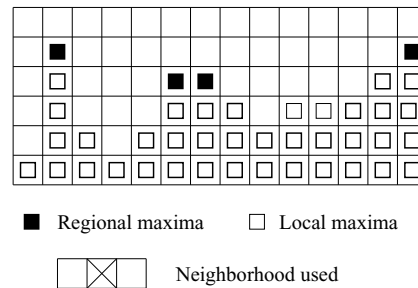


Figure 13.31: 1D illustration of regional and local maxima. © Cengage Learning 2015.

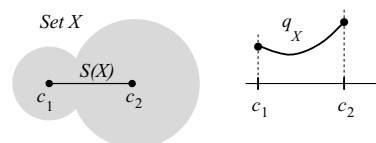


Figure 13.32: Skeleton of a set X , and associated quench function $q_X(p)$. Regional maxima give the ultimate erosion. © Cengage Learning 2015.

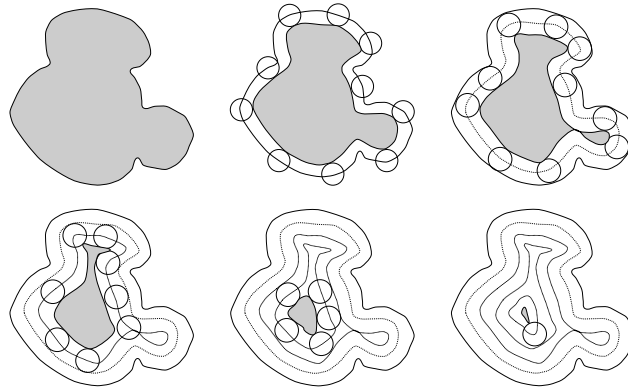


Figure 13.33: When successively eroded, components are separated from the rest and finally disappear from the image. The union of residua just before disappearance gives ultimate erosion.
© Cengage Learning 2015.

13.5.5 Ultimate erosion and distance functions

We seek to present the ultimate erosion procedure formally, and introduce the **morphological reconstruction** operator for this purpose. Assume two sets A , B , $B \subseteq A$. The reconstruction $\rho_A(B)$ of the set A from set B is the union of connected components of A with non-empty intersection with B (see Figure 13.35—notice that set A consists of two components). Notice that B may typically consist of markers that permit the reconstruction of the required part of the set A . Markers point to the pixel or small region that belongs to the object. Markers are found either by a human operator (which is common, e.g. in biology) or can be found automatically. Morphological reconstruction will be discussed in detail in Section 13.5.7.

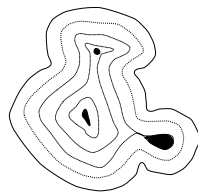


Figure 13.34: Ultimate erosion is the union of residual connected components before they disappear during erosions.
© Cengage Learning 2015.

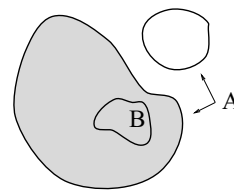


Figure 13.35: Reconstruction $\rho_A(B)$ (in gray) of the set A from B . Notice that set A may consist of more than one connected component. Note that only the portion of A that contains B is reconstructed while the other component of A that does not include B will disappear after reconstruction of A from B . © Cengage Learning 2015.

Let $\mathcal{N}$ be the set of integers. Ultimate erosion can be expressed by the formula

$$\text{Ult}(X) = \bigcup_{n \in \mathcal{N}} \left((X \ominus nB) \setminus \rho_{X \ominus nB}(X \ominus (n+1)B) \right). \quad (13.54)$$

There is a computationally effective ultimate erosion algorithm that uses the distance function (which is the core of several other quick morphological algorithms, as we shall see). The **distance function** $\text{dist}_X(p)$ associated with each pixel p of the set X is the size of the first erosion of X that does not contain p , i.e.,

$$\forall p \in X, \quad \text{dist}_X(p) = \min \{n \in \mathcal{N}, p \text{ not in } (X \ominus nB)\}. \quad (13.55)$$

This behaves as one would expect: $\text{dist}_X(p)$ is the shortest distance between the pixel p and background X^C .

There are two direct applications of the distance function.

- The ultimate erosion of a set X corresponds to the union of the regional maxima of the distance function of X .
- The skeleton by maximal balls of a set X corresponds to the set of local maxima of the distance function of X .

The final concept to be introduced is **skeleton by influence zones**, often abbreviated *SKIZ*. Let X be composed of n connected components X_i , $i = 1, \dots, n$. The influence zone $Z(X_i)$ consists of points which are closer to set X_i than to any other connected component of X :

$$Z(X_i) = \{p \in \mathcal{Z}^2, \forall i \neq j, d(p, X_i) \leq d(p, X_j)\}. \quad (13.56)$$

The skeleton by influence zones denoted $\text{SKIZ}(X)$ is the set of boundaries of influence zones $\{Z(X_i)\}$.

13.5.6 Geodesic transformations

Geodesic methods [Vincent, 1995] modify morphological transformations to operate only on some part of an image. For instance, if an object is to be reconstructed from a marker, say a nucleus of a cell, it is desirable to avoid growing from a marker outside the cell. Another important advantage of geodesic transformations is that the structuring element can vary at each pixel, according to the image.

The basic concept of geodesic methods in morphology is geodesic distance. The path between two points is constrained within some set. The term has its roots in an old discipline—geodesy—that measures distances on the Earth's surface. Suppose that a traveler seeks the distance between London and Tokyo—the shortest distance passes *through* the Earth, but obviously the geodesic distance that is of interest to the traveler is constrained to the Earth's surface.

The **geodesic distance** $d_X(x, y)$ is the shortest path between two points x, y which remains entirely contained in the set X . If there is no path connecting points x, y , we set $d_X(x, y) = +\infty$. Geodesic distance is illustrated in Figure 13.36.

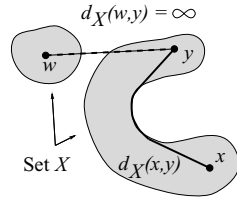


Figure 13.36: Geodesic distance $d_X(x, y)$.
© Cengage Learning 2015.

A **geodesic ball** is a ball constrained by some set X . The geodesic ball $B_X(p, n)$ of center $p \in X$ and radius n is defined as

$$B_X(p, n) = \{p' \in X, d_X(p, p') \leq n\}. \quad (13.57)$$

The existence of a geodesic ball permits dilation or erosion only within some subset of the image; this leads to definitions of geodesic dilations and erosions of a subset Y of X .

The **geodesic dilation** $\delta_X^{(n)}$ of size n of a set Y inside the set X is defined as

$$\delta_X^{(n)}(Y) = \bigcup_{p \in Y} B_X(p, n) = \{p' \in X, \exists p \in Y, d_X(p, p') \leq n\}. \quad (13.58)$$

Similarly the dual operation of **geodesic erosion** $\epsilon_X^{(n)}(Y)$ of size n of a set Y inside the set X can be written as

$$\epsilon_X^{(n)}(Y) = \{p \in Y, B_X(p, n) \subseteq Y\} = \{p \in Y, \forall p' \in X \setminus Y, d_X(p, p') > n\}. \quad (13.59)$$

Geodesic dilation and erosion are illustrated in Figure 13.37.

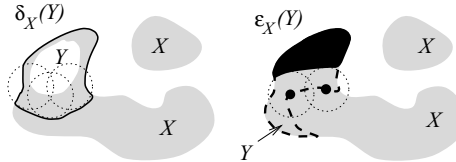


Figure 13.37: Geodesic dilation (left) and erosion (right) of the set Y inside the set X . © Cengage Learning 2015.

The outcome of a geodesic operation on a set $Y \subseteq X$ is always included within the set X . Regarding implementation, the simplest geodesic dilation of size 1 ($\delta_X^{(1)}$) of a set Y inside X is obtained as the intersection of the unit-size dilation of Y (with respect to the unit ball B) with the set X

$$\delta_X^{(1)} = (Y \oplus B) \cap X. \quad (13.60)$$

Larger geodesic dilations are obtained by iteratively composing unit dilations n times

$$\delta_X^{(n)} = \underbrace{\delta_X^{(1)} \left(\delta_X^{(1)} \left(\delta_X^{(1)} \dots \left(\delta_X^{(1)} \right) \right) \right)}_{n \text{ times}}. \quad (13.61)$$

The fast iterative way to calculate geodesic erosion is similar.

13.5.7 Morphological reconstruction

Assume that we want to reconstruct objects of a given shape from a binary image that was originally obtained by thresholding. All connected components in the input image constitute the set X . However, only some of the connected components were marked by markers that represent the set Y . This task and its desired result are shown in Figure 13.38.

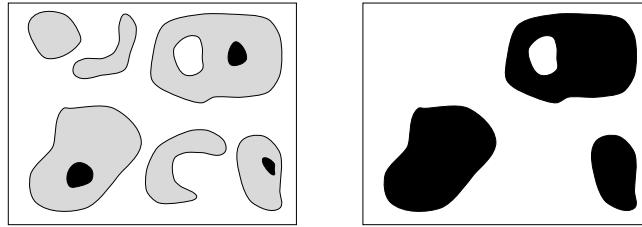


Figure 13.38: Reconstruction of X (X shown in light gray) from markers Y (black). The reconstructed result is shown in black on the right side. © Cengage Learning 2015.

Successive geodesic dilations of the set Y inside the set X enable the reconstruction of the connected components of X that were initially marked by Y . When dilating from the marker, it is impossible to intersect a connected component of X which did not initially contain a marker Y ; such components disappear.

Geodesic dilations terminate when all connected components set X previously marked by Y are reconstructed, i.e., idempotency is reached:

$$\forall n > n_0, \delta_X^{(n)}(Y) = \delta_X^{(n_0)}(Y). \quad (13.62)$$

This operation is called **reconstruction** and denoted by $\rho_X(Y)$. Formally,

$$\rho_X(Y) = \lim_{n \rightarrow \infty} \delta_X^{(n)}(Y). \quad (13.63)$$

In some applications it is desirable that one connected component of X is marked by several markers Y . If it is not acceptable for the sets grown from various markers to become connected, the notion of influence zones can be generalized to **geodesic influence zones** of the connected components of set Y inside X . This is illustrated in Figure 13.39.

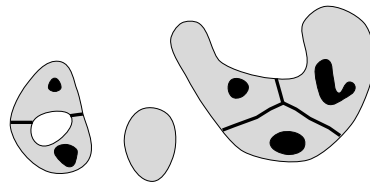


Figure 13.39: Geodesic influence zones. © Cengage Learning 2015.

We are now ready to generalize the reconstruction to gray-scale images; this requires the extension of geodesy to gray-scale images. The core of the extension is the statement (which is valid for discrete images) that any increasing transformation defined for binary

images can be extended to gray-level images [Serra, 1982]. By this transformation we mean a transformation Ψ such that

$$\forall X, Y \subset \mathcal{Z}^2, Y \subseteq X \implies \Psi(Y) \subseteq \Psi(X). \quad (13.64)$$

The generalization of transformation Ψ is achieved by viewing a gray-level image I as a stack of binary images obtained by successive thresholding—this is called the threshold decomposition of image I [Maragos and Ziff, 1990]. Let D_I be the domain of the image I , and the gray values of image I be in $\{0, 1, \dots, N\}$. The thresholded images $T_k(I)$ are

$$T_k(I) = \{p \in D_I, I(p) \geq k\}, \quad k = 0, \dots, N. \quad (13.65)$$

The idea of threshold decomposition is illustrated in Figure 13.40.

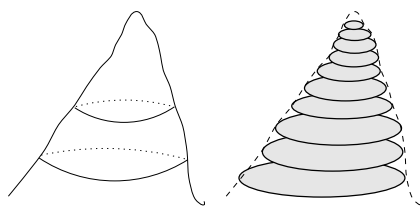


Figure 13.40: Threshold decomposition of a gray-scale image. © Cengage Learning 2015.

Threshold-decomposed images $T_k(I)$ obey the inclusion relation

$$\forall k \in [1, N], T_k(I) \subseteq T_{k-1}(I). \quad (13.66)$$

Consider the increasing transformation Ψ applied to each threshold-decomposed image; their inclusion relationship is kept. The transformation Ψ can be extended to gray-scale images using the following **threshold decomposition principle**:

$$\forall p \in D_I, \Psi(I)(p) = \max \{k \in [0, \dots, N], p \in \Psi(T_k(I))\}. \quad (13.67)$$

Returning to the reconstruction transformation, the binary geodesic reconstruction ρ is an increasing transformation, as it satisfies

$$Y_1 \subseteq Y_2, X_1 \subseteq X_2, Y_1 \subseteq X_1, Y_2 \subseteq X_2 \implies \rho_{X_1}(Y_1) \subseteq \rho_{X_2}(Y_2). \quad (13.68)$$

We are ready to generalize binary reconstruction to **gray-level reconstruction** applying the threshold decomposition principle (equation 13.67). Let J, I be two gray-scale images defined on the same domain D , with gray-level values from the discrete interval $[0, 1, \dots, N]$. If, for each pixel $p \in D$, $J(p) \leq I(p)$, the gray-scale reconstruction $\rho_I(J)$ of image I from image J is given by

$$\forall p \in D, \rho_I(J)(p) = \max \{k \in [0, N], p \in \rho_{T_k}(T_k(J))\}. \quad (13.69)$$

Recall that binary reconstruction grows those connected components of the mask which are marked. The gray-scale reconstruction extracts peaks of the mask I that are marked by J (see Figure 13.41).

The duality between dilation and erosion permits the expression of gray-scale reconstruction using erosion.

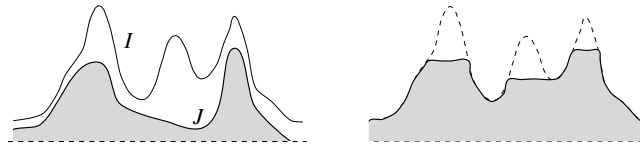


Figure 13.41: Gray-scale reconstruction of mask I from marker J . © Cengage Learning 2015.

13.6 Granulometry

Granulometry was introduced by stereologists (mathematicians attempting to understand 3D shape from cross sections)—the name comes from the Latin **granulum**, meaning grain. Matheron [Matheron, 1967] used it as a tool for studying porous materials, where the distribution of pore sizes was quantified by a sequence of openings of increasing size. Currently, granulometry is an important tool of mathematical morphology, particularly in material science and biology applications. The main advantage is that granulometry permits the extraction of shape information without a priori segmentation.

Consider first a sieving analysis analogy, where the input is a heap of stones (or granules) of different sizes—the task is to analyze how many stones in the heap fit into several size classes. Such a task is solved by sieving using several sieves with increasing sizes of holes in the mesh. The result of analysis is a discrete function; on its horizontal axis are increasing sizes of stones and on its vertical axis the numbers of stones of that size. In morphological granulometry, this function is called a **granulometric spectrum** or **granulometric curve**.

In binary morphology, the task is to calculate a granulometric curve where the independent variable is the size of objects in the image. The value of the granulometric curve is the number of objects of given size in the image. The most common approach is that sieves with increasing hole sizes (as in the example) are replaced by a sequence of openings with structural elements of increasing size.

Granulometry plays a very significant role in mathematical morphology, analogous to the role of frequency analysis in image processing or signal analysis. Frequency analysis expands a signal into a linear combination of harmonic signals of growing frequency. The frequency spectrum provides the contribution of individual harmonic signals—it is clear that the granulometric curve (spectrum) is analogous to a frequency spectrum.

Let $\Psi = (\psi_\lambda)$, $\lambda \geq 0$, be a family of transformations depending on a parameter λ . This family constitutes a **granulometry** if and only if the following properties of the transformation ψ hold:

$$\begin{aligned} \forall \lambda \geq 0 \quad & \psi_\lambda \text{ is increasing,} \\ & \psi_\lambda \text{ is anti-extensive,} \\ \forall \lambda \geq 0, \mu \geq 0 \quad & \psi_\lambda \psi_\mu = \psi_\mu \psi_\lambda = \psi_{\max(\lambda, \mu)}. \end{aligned} \quad (13.70)$$

The consequence of property (equation 13.70) is that for every $\lambda \geq 0$ the transformation ψ_λ is idempotent. (ψ_λ) , $\lambda \geq 0$ is a decreasing family of openings (more precisely, algebraic openings [Serra, 1982] that generalize the notion of opening presented earlier). It can be shown that for any convex structuring element B , the *family of openings* with respect to $\lambda B = \{\lambda b, b \in B\}$, $\lambda \geq 0$, constitutes a granulometry [Matheron, 1975].

Consider more intuitive granulometry acting on discrete binary images (i.e., sets). Here the granulometry is a sequence of openings ψ_n indexed by an integer $n \geq 0$ —

each opening result is smaller than the previous one. Recall the analogy with sieving analysis; each opening, which corresponds to one sieve mesh size, removes from the image more than the previous one. Finally, the empty set is reached. Each sieving step is characterized by some measure $m(X)$ of the set (image) X (e.g., number of pixels in a 2D image, or volume in 3D). The rate at which the set is sieved characterizes the set. The pattern spectrum provides such a characteristic.

The **pattern spectrum**, also called *granulometric curve*, of a set X with respect to the granulometry $\Psi = \psi_n, n \geq 0$ is the mapping

$$PS_{\Psi}(X)(n) = m[\psi_n(X)] - m[\psi_{n-1}(X)], \quad \forall n > 0. \quad (13.71)$$

The sequence of openings $\Psi(X), n \geq 0$ is a decreasing sequence of sets, i.e., $[\psi_0(X) \supseteq \psi_1(X) \supseteq \psi_2(X) \supseteq \dots]$. The granulometry and granulometric curve can be used.

Suppose that the granulometric analysis with family of openings needs to be computed for a binary input image. The binary input image is converted into a gray-level image using a granulometry function $G_{\Psi}(X)$, and the pattern spectrum PS_{Ψ} is calculated as a histogram of the granulometry function.

The *granulometry function* $G_{\Psi}(X)$ of a binary image X from granulometry $\Psi = (\psi_n)$, $n \geq 0$, maps each pixel $x \in X$ to the size of the first n such that $x \notin \psi_n(X)$:

$$x \in X, G_{\Psi}(X)(x) = \min \{n > 0, x \notin \psi_n(X)\}. \quad (13.72)$$

The pattern spectrum PS_{Ψ} of a binary image X for granulometry $\Psi = (\psi_n), n \geq 0$, can be computed from the granulometry function $G_{\Psi}(X)$ as its histogram

$$\forall n > 0, PS_{\Psi}(X)(n) = \text{card}\{p, G_{\Psi}(X)(p) = n\}, \quad (13.73)$$

where ‘card’ denotes cardinality. An example of granulometry is given in Figure 13.42. The input binary image with circles of different radii is shown in Figure 13.42a; Figure 13.42b shows one of the openings with a square structuring element. Figure 13.42c

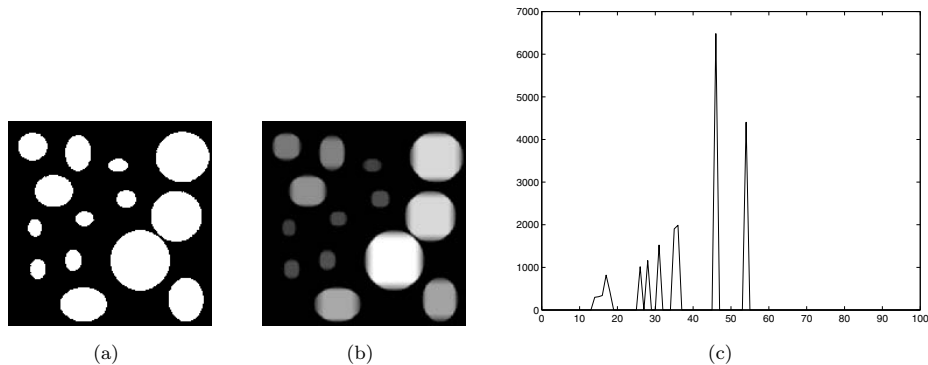


Figure 13.42: Example of binary granulometry performance. (a) Original binary image. (b) Maximal square probes inscribed—the initial probe size was 2×2 pixels. (c) Granulometric power spectrum as histogram of (b)—the horizontal axis gives the size of the object and the vertical axis the number of pixels in an object of given size. *Courtesy of P. Kodl, Rockwell Automation Research Center, Prague.*

illustrates the granulometric power spectrum. At a coarse scale, three most significant signals in the power spectrum indicate three prevalent sizes of object. The less significant signals on the left side are caused by the artifacts that occur due to discretization. The Euclidean circles have to be replaced by digital entities (squares).

Notice that granulometries extract size information without the need to identify (segment) each object a priori. In applications, this is used for shape description, feature extraction, texture classification, and removal of noise introduced by image borders.

Granulometric analysis was historically costly to perform but this is no longer the case. For binary images, the basic idea towards speed-up is to use linear structuring elements for openings and more complex 2D ones derived from it, such as cross, square, or diamond (see Figure 13.43). The next source of computational saving is the fact that some 2D structuring elements can be decomposed as Minkowski addition of two 1D structuring elements. For example, the square structuring element can be expressed as Minkowski addition of horizontal and vertical lines [Haralick et al., 1995; Vincent, 1995]. *Gray-scale granulometric analysis* permits the extraction of size information directly from gray-level images. The interested reader should consult [Vincent, 1994].

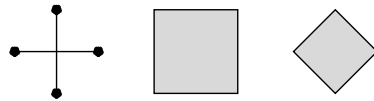


Figure 13.43: Structural elements used for fast binary granulometry are derived from line structuring elements, e.g., cross, square, and diamond. © Cengage Learning 2015.

13.7 Morphological segmentation and watersheds

13.7.1 Particle segmentation, marking, and watersheds

Segmentation usually means finding objects of interest in an image. Mathematical morphology helps mainly to segment images of texture or images of particles—here we consider **particle segmentation** in which the input image can be either binary or gray-scale. In the binary case, the task is to segment overlapping particles; in the gray-scale case, the segmentation is the same as object contour extraction [Vincent, 1995].

Morphological particle segmentation is performed in two basic steps: (1) location of particle markers, and (2) watersheds used for particle reconstruction. The latter is explained later in this section.

Marker extraction resembles human behavior when one is asked to indicate objects; the person just points to objects and does not outline boundaries. The **marker** of an object or set X is a set M that is included in X . Markers have the same homotopy as the set X , and are typically located in the central part of the object (particle).

A robust marker-finding technique will need to know the nature of the objects sought, and thus application-specific knowledge should be used. Often combinations of non-morphological and morphological approaches are used. Moreover, object marking is in many cases left to the user, who marks objects manually on the screen. Typically, software for analysis of microscopic images has user-friendly tools for manual or semi-automatic marking.

When objects are marked, they can be grown from the markers, e.g., using the watershed transformation (Section 13.7.3), which is motivated by the topographic view

of images. Consider the analogy of a landscape and rain; water will find the swiftest descent path until it reaches some lake or sea. We know that lakes and seas correspond to regional minima. The landscape can be entirely partitioned into regions which attract water to a particular sea or lake—we call these **catchment basins**. These regions are influence zones of the regional minima in the image. **Watersheds** (or **watershed lines**) separate catchment basins. Watersheds and catchment basins are illustrated in Figure 13.44.

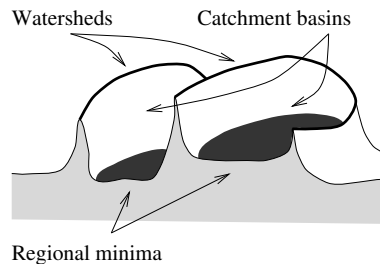


Figure 13.44: Illustration of catchment basins and watersheds in a 3D landscape view.
© Cengage Learning 2015.

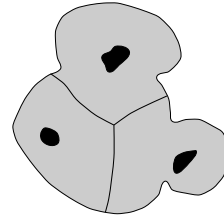


Figure 13.45: Segmentation by geodesic influence zones (SKIZ) need not lead to correct results. © Cengage Learning 2015.

13.7.2 Binary morphological segmentation

If the task is to find objects that differ in brightness from an uneven background, the top hat transformation (Section 13.5.4) is a simple solution. This approach finds peaks in the image function that differ from the local background: the gray-level shape of the peaks does not play any role, but the shape of the structuring element does. Watershed segmentation takes into account both sources of information and supersedes the top hat method.

Morphological segmentation in binary images aims to find regions corresponding to individual overlapping objects (typically particles), and most of the tools for performing this task have already been explained. Each particle is marked first—ultimate erosion may be used for this purpose (Section 13.5.4), or markers may be placed manually. The next task is to grow objects from the markers provided they are kept within the limits of the original set and parts of objects are not joined when they come close to each other.

The oldest technique for this purpose is called **conditional dilation**. Ordinary dilation is used for growing, and the result is constrained by two conditions: remain in the original set, and do not join particles.

Geodesic reconstruction (Section 13.5.7) is more sophisticated and performs much faster than conditional dilation. The structuring element adapts according to the neighborhood of the processed pixel.

Geodesic influence zones (Section 13.5.7) are sometimes used for segmenting particles. Figure 13.45 shows that the result can differ from our intuitive expectation.

The best solution is the **watershed transformation**; for the theoretical background and fast implementation techniques, see Section 13.7.3 and [Bleau et al., 1992; Vincent, 1993, 1995]. The original binary image is converted into gray-scale using the negative distance transform (equation 13.55). If a drop of water falls onto a topographic surface

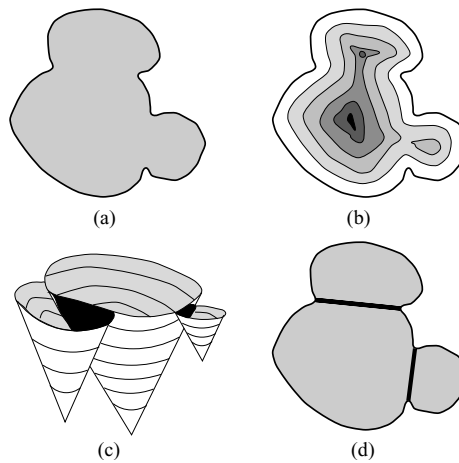


Figure 13.46: Segmentation of binary particles. (a) Input binary image. (b) Gray-scale image created from (a) using the $-dist$ function. (c) Topographic notion of the catchment basin. (d) Correctly segmented particles using watersheds of image (b). © Cengage Learning 2015.

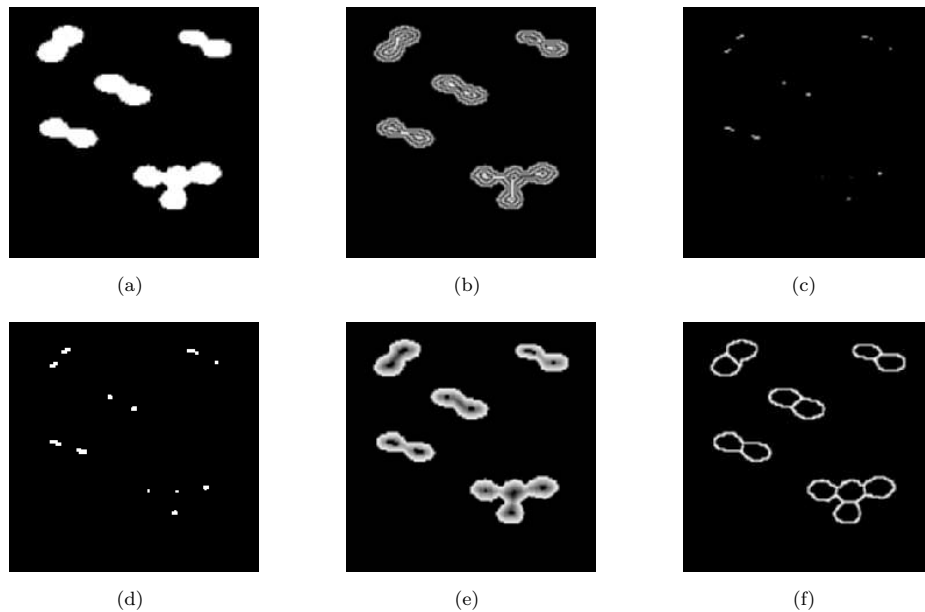


Figure 13.47: Particle segmentation by watersheds. (a) Original binary image. (b) Distance function visualized using contours. (c) Regional maxima of the distance function used as particle markers. (d) Dilated markers. (e) Inverse of the distance function with the markers superimposed. (f) Resulting contours of particles obtained by watershed segmentation. Courtesy of P. Kodl, Rockwell Automation Research Center, Prague.

of this image, it will follow the steepest slope towards a regional minimum. This idea is illustrated in Figure 13.46.

Application of watershed particle segmentation is shown in Figure 13.47. An image of a few touching particles is the input Figure 13.47a, and the distance function calculated from the background is visualized using contours in Figure 13.47b. Regional maxima of the distance function serve as markers of the individual particles (Figure 13.47c). The markers are dilated in Figure 13.47d. In preparation for watershed segmentation, the distance function is negated, and is shown together with the dilated markers in Figure 13.47e. The final result of particle separation is illustrated in Figure 13.47f, where particle contours are shown.

13.7.3 Gray-scale segmentation, watersheds

The markers and watersheds method can also be applied to gray-scale segmentation. Watersheds are also used as crest-line extractors in gray-scale images. The contour of a region in a gray-level image corresponds to points in the image where gray-levels change most quickly—this is analogous to edge-based segmentation considered in Chapter 6. The watershed transformation is applied to the gradient magnitude image in this case (see Figure 13.48). There is a simple approximation to the gradient image used in mathematical morphology called Beucher's gradient [Serra, 1982], calculated as the algebraic difference of unit-size dilation and unit-size erosion of the input image X

$$\text{grad}(X) = (X \oplus B) - (X \ominus B). \quad (13.74)$$

The main problem with segmentation via gradient images without markers is over-segmentation (Figure 13.47c). Some techniques to limit this in watershed segmentation are given in [Vincent, 1993]. Watershed segmentation methods with markers do not suffer from over-segmentation, of course.

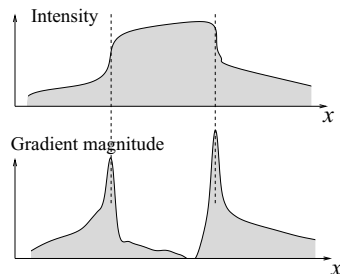


Figure 13.48: Segmentation in gray-scale images using gradient magnitude. © Cengage Learning 2015.

An example from ophthalmology illustrates the application of watershed segmentation. The input image shows a microscopic picture of part of a human retina, Figure 13.49a—the task is to segment individual cells on the retina. The markers/watershed paradigm was followed, with markers being found using a carefully tuned Gaussian filter (see Figure 13.49b). The final result with outlined contours of cells is in Figure 13.49c.

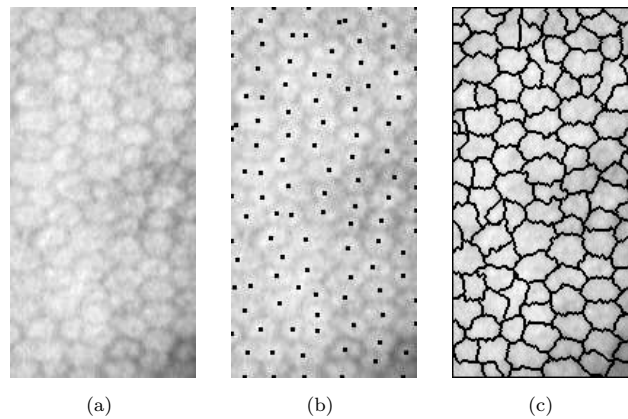


Figure 13.49: Watershed segmentation of a human retina image. (a) Original gray-scale image. (b) Dots are superimposed markers found by non-morphological methods. (c) Boundaries of retina cells found by watersheds from markers (b). *Courtesy of R. Šára, Czech Technical University, Prague and P. Kodl, Rockwell Automation Research Center Prague.*

13.8 Summary

- **Mathematical morphology**

- Mathematical morphology stresses the role of *shape* in image pre-processing, segmentation, and object description. It leads to mathematically sound and fast algorithms.
- The basic entity is a *point set*; it operates using transformations described using operators in a relatively simple *non-linear algebra*. It is a counterpart to traditional signal processing based on linear operators.
- It is divided into *binary mathematical morphology* which operates on binary images, and *gray-level mathematical morphology* which acts on gray-level images.

- **Morphological operations**

- Morphological operations are *relations of two sets*. One is an image and the second a small *structuring element* that systematically traverses the image. Its relation to the image in each position is stored in the output image.
- Fundamental operations of mathematical morphology include *dilation* which expands objects and *erosion* which shrinks them. They are not invertible; their combination constitutes new operations—*opening* and *closing*.
- Thin and elongated objects are often simplified using a *skeleton*—a line that is in ‘the middle of the object’.
- The *distance function* to the background constitutes a basis for many fast operations. *Ultimate erosion* is used to mark blob centers. There is an efficient reconstruction algorithm that grows the object from the marker to its original boundary.

718 Chapter 13: Mathematical morphology

- *Geodesic transformations* allow changes to the structuring element during processing and thus provide more flexibility.
- They provide quick and efficient segmentation algorithms: the *watershed transform* is one of the better ones. Boundaries of regions are influence zones of regional minima (i.e., seas and lakes in the landscape). Region boundaries are watershed lines between these seas and lakes.
- Segmentation is often performed from interactively chosen *markers*, or from some automatic procedure that takes into account semantic properties of the image.
- **Granulometry** is a quantitative tool for analyzing images with particles of different size (similar to sieving analysis). The result is a discrete *granulometric curve* (spectrum).

13.9 Exercises

Short-answer questions

- S13.1 What is mathematical morphology?
- S13.2 What is a structuring element? What is its role in mathematical morphology?
- S13.3 Give the definition of erosion and dilation for binary images.
- S13.4 Is erosion a commutative operation?
- S13.5 What is the difference between the result of opening performed once and twice? What is idempotency?
- S13.6 Gray-level dilation and erosion are a generalization of the same notions in binary images. Describe how they are done.
- S13.7 What is the top hat transformation and when is it used? Sketch a 1D example of the image profile demonstrating the use of top hat transform, show the top-hat transform result.
- S13.8 What is the skeleton by maximal balls? What does the skeleton of two touching filled circles look like?
- S13.9 What does the homotopic skeleton of two touching balls look like?
- S13.10 Explain the role of ultimate erosion for marking particles.
- S13.11 What is a regional maximum?
- S13.12 What is geodesic distance? How is it used in mathematical morphology?
- S13.13 What is granulometry?
- S13.14 Name three application areas in which granulometry analysis may be used.
- S13.15 What is the relation between a granulometric curve (spectrum) and the Fourier spectrum?
- S13.16 What is a watershed? How is it used for morphological segmentation?

Problems

- P13.1 It is probably not going to be computationally efficient to re-implement morphological algorithms personally. There are many packages around that perform morphology fast—Matlab is just one example. Locate such software and exercise any or all of the algorithms described in this chapter.
- P13.2 Prove that dilation is commutative and associative.
- P13.3 Opening followed by closing can be used to remove salt and pepper noise from binary images. Explain the principle and discuss the shape and size of a structuring element suitable for this operation.
- P13.4 Consider a one-dimensional gray-scale image (signal). Draw a picture that demonstrates that gray-scale dilation fills narrow bays in the image.
- P13.5 Explain how the top hat transformation permits the segmentation of dark characters on a light background with varying intensity. Draw a picture of a one-dimensional cross section through the image.
- P13.6 What is a homotopic transformation? Describe at least two algorithms for skeletonization. Do they produce a homotopic skeleton?
- P13.7 What is the difference between thinning with the Golay structuring elements L (equation 13.51) and E (equation 13.52)?
- P13.8 Explain the calculation steps of the skeleton by maximal balls in Figure 13.29.
- P13.9 Explain the role of markers in morphological segmentation. Why would an attempt to perform watershed segmentation without markers lead to over-segmentation?
- P13.10 Dilate the image given in Figure 13.50a with the structuring element given in Figure 13.50b.

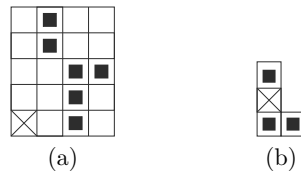


Figure 13.50: (a) Image to be processed. Assume that image data are undefined outside of the image domain. (b) Structuring element.

© Cengage Learning 2015.

- P13.11 Using ultimate erosion, develop an approach for counting adjacent/touching objects in an image (like pebbles or coffee beans). Demonstrate robustness of your approach (or lack of) with respect to the initial segmentation of the objects (binarization). Apply the same approach to images of objects of increasing segmentation difficulty.
- P13.12 Assuming that the approach developed in Problem P13.11 does not perform sufficiently well for some of the more difficult images, augment your method with other morphologic steps. Use *only* mathematical morphology operations to solve this problem.
- P13.13 Use the markers obtained by ultimate erosion (Problems P13.11, P13.12) to segment the same images using watershed transformation.
- P13.14 Modify the watershed algorithm so that objects and borders are segmented separately. (Hint: apart from extracting object markers, it is necessary to extract markers for borders.) Use *only* mathematical morphology operations to solve this problem.

- P13.15 Make yourself familiar with solved problems and **Matlab** implementations of selected algorithms provided in the corresponding chapter of the **Matlab Companion** to this text [Svoboda et al., 2008]. The **Matlab Companion** homepage <http://visionbook.felk.cvut.cz> offers images used in the problems, and well-commented **Matlab** code is provided for educational purposes.
- P13.16 Use the **Matlab Companion** [Svoboda et al., 2008] to develop solutions to additional exercises and practical problems provided there. Implement your solutions using **Matlab** or other suitable programming languages.

13.10 References

- Bleau A., deGuise J., and LeBlanc R. A new set of fast algorithms for mathematical morphology II. Identification of topographic features on grayscale images. *CVGIP: Image Understanding*, 56(2):210–229, 1992.
- Blum H. A transformation for extracting new descriptors of shape. In Wathen-Dunn W., editor, *Proceedings of the Symposium on Models for the Perception of Speech and Visual Form*, pages 362–380, Cambridge, MA, 1967. MIT Press.
- Dougherty E. R. *An Introduction to Mathematical Morphology Processing*. SPIE Press, Bellingham, WA, 1992.
- Giardina C. R. and Dougherty E. R. *Morphological Methods in Image and Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1988.
- Golay M. J. E. Hexagonal parallel pattern transformation. *IEEE Transactions on Computers*, C-18:733–740, 1969.
- Haralick R. M. and Shapiro L. G. *Computer and Robot Vision, Volume I*. Addison-Wesley, Reading, MA, 1992.
- Haralick R. M., Stenberg S. R., and Zhuang X. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):532–550, 1987.
- Haralick R. M., Katz P. L., and Dougherty E. R. Model-based morphology: The opening spectrum. *Graphical Models and Image Processing*, 57(1):1–12, 1995.
- Heijmans H. J. *Morphological Image Operators*. Academic Press, Boston, 1994.
- Ma C. M. and Sonka M. A fully parallel 3D thinning algorithm and its applications. *Computer Vision and Image Understanding*, 64:420–433, 1996.
- Maragos P. and Ziff R. Threshold superposition in morphological image analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5), 1990.
- Matheron G. *Eléments pour une theorie des milieux poreux* (in French). Masson, Paris, 1967.
- Matheron G. *Random Sets and Integral Geometry*. Wiley, New York, 1975.
- Meyer F. Contrast feature extraction. In Chermant J.-L., editor, *Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*, Stuttgart, Germany, 1978. Riederer Verlag. Special issue of *Practical Metallography*.
- Palagyi K., Tschirren J., Hoffman E. A., and Sonka M. Quantitative analysis of pulmonary airway tree structures. *Computers in Biology and Medicine*, 36:974–996, 2006.
- Serra J. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.
- Serra J. Morphological optics. *Journal of Microscopy*, 145(1):1–22, 1987.
- Soille P. *Morphological Image Analysis*. Springer-Verlag, Berlin, 2 edition, 2003.

- Svoboda T., Kybic J., and Hlavac V. *Image Processing, Analysis, and Machine Vision: A MATLAB Companion*. Thomson Engineering, 2008.
- Vincent L. Fast opening functions and morphological granulometries. In *Proceedings Image Algebra and Morphological Image Processing V*, pages 253–267, San Diego, CA, 1994. SPIE.
- Vincent L. Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, 1993.
- Vincent L. Efficient computation of various types of skeletons. In *Proceedings of the SPIE Symposium Medical Imaging*, San Jose, CA, volume 1445, Bellingham, WA, 1991. SPIE.
- Vincent L. Lecture notes on granulometries, segmentation and morphological algorithms. In Wojciechowski K., editor, *Proceedings of the Summer School on Morphological Image and Signal Processing*, Zakopane, Poland, pages 119–216. Silesian Technical University, Gliwice, Poland, 1995.

Chapter 14

Image data compression

Image processing is often very difficult because of the large amounts of data used to represent an image. Technology permits ever-increasing image resolution (spatially and in gray-levels), and increasing numbers of spectral bands, and there is a consequent need to limit the resulting data volume. Examples of this problem are everywhere; raw video—whether high definition or not—represents many megabytes per second. Medical applications such as CT and MR are 3D (or 4D, if dynamic in time) and are likewise extremely data hungry. One possible approach to decreasing the necessary amount of storage is to work with compressed image data.

We have seen that segmentation techniques have the side effect of image compression; by removing all areas and features that are not of interest, and leaving only boundaries or region descriptors, the reduction in data quantity is considerable. However, from this sort of representation no image reconstruction (or only very limited reconstruction) to the original image is possible. Conversely, image compression algorithms aim to remove redundancy in data in a way which makes image reconstruction possible; this is sometimes called **information preserving compression**. Compression is the main goal of the algorithm—we aim to represent an image using fewer bits per pixel. It is necessary to find statistical properties of the image to design an appropriate compression transformation of the image; the more correlated the image data are, the more data items can be removed. In this chapter, we will discuss this group of methods which do not change image entropy or image information content; the literature on compression is compendious—just three useful references would be [Salomon, 2000; Sayood, 2000; Shukla and Prasad, 2011].

A general algorithm for data compression and image reconstruction is shown in a block diagram in Figure 14.1. The first step removes information redundancy caused by high correlation of image data—transform compressions, predictive compressions, and hybrid approaches are used. The second step is coding of transformed data using a code of fixed or variable-length. An advantage of variable-length codes is the possibility of coding more frequent data using shorter code words and therefore increasing compression efficiency, while an advantage of fixed length coding is a standard codeword length that offers easy handling and fast processing. Compressed data are decoded after transmission

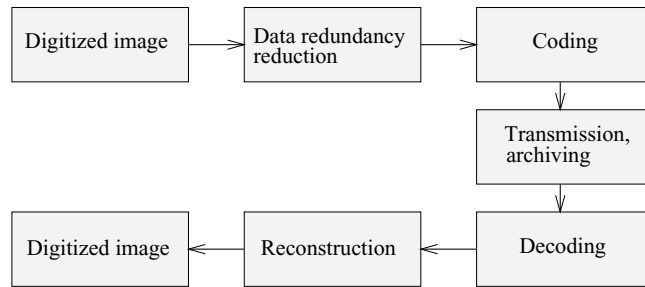


Figure 14.1: Data compression and image reconstruction. © Cengage Learning 2015.

or archiving and reconstructed. Note that no non-redundant image data may be lost in the data compression process—otherwise error-free reconstruction is impossible.

Data compression methods can be divided into two principal groups: information preserving—or **lossless**—compression permits error-free data reconstruction, while **lossy** compression methods do not preserve information completely. In image processing, a faithful reconstruction is often not necessary in practice and then the requirements are weaker, but the compression must not cause significant changes in an image. Data compression success in the reconstructed image is usually measured by the mean square error (MSE), signal-to-noise ratio etc., although these global error measures do not always reflect subjective image quality.

Compression design consists of two parts. Image data properties must be determined first; gray-level histograms, image entropy, various correlation functions, etc., often serve this purpose. The second part yields an appropriate compression technique with respect to measured image properties.

Data compression methods with loss of information are typical in image processing and therefore this group of methods is described in some detail. Although lossy compression techniques can give substantial image compression with very good quality reconstruction, there are considerations that may prohibit their use. For example, diagnosis in medical imaging is often based on visual image inspection, so no loss of information can be tolerated and information preserving techniques must be applied. Information preserving compression methods are mentioned briefly at the end of the chapter.

14.1 Image data properties

Information content of an image is an important property, of which *entropy* is a measure (Section 2.3.3). If an image has G gray-levels and the probability of gray-level k is $P(k)$ (see Section 3.3), then entropy H_e , not considering correlation of gray-levels, is defined as

$$H_e = - \sum_{k=0}^{G-1} P(k) \log_2 (P(k)) . \quad (14.1)$$

Information **redundancy** r is defined as

$$r = b - H_e , \quad (14.2)$$

where b is the smallest number of bits with which the image quantization levels can be represented. This definition of image information redundancy can be evaluated only if a good estimate of entropy is available, which is usually not so because the necessary statistical properties of the image are not known. Image data entropy however can be estimated from a gray-level histogram [Moik, 1980; Pratt, 1991]. Let $h(k)$ be the frequency of gray-level k in an image f , $0 \leq k \leq 2^b - 1$, and let the image size be $M \times N$. The probability of occurrence of gray-level k can be estimated as

$$\tilde{P}(k) = \frac{h(k)}{MN} \quad (14.3)$$

and the entropy can be estimated as

$$\tilde{H}_e = - \sum_{k=0}^{2^b-1} \tilde{P}(k) \log_2 (\tilde{P}(k)) . \quad (14.4)$$

The information redundancy estimate is $\tilde{r} = b - \tilde{H}_e$. The definition of the **compression ratio** K is then

$$K = \frac{b}{\tilde{H}_e} . \quad (14.5)$$

Note that a gray-level histogram gives an inaccurate estimate of entropy because of gray-level correlation. A more accurate estimate can be obtained from a histogram of the first gray-level differences.

Theoretical limits of possible image compression can be found using these formulae. For example, the entropy of satellite remote sensing data may be $\tilde{H}_e \in [4, 5]$, where image data are quantized into 256 gray-levels, or 8 bits per pixel. We can easily compute the information redundancy as $\tilde{r} \in [3, 4]$ bits. This implies that these data can be represented by an average data volume of 4–5 bits per pixel with no loss of information, and the compression ratio would be $K \in [1.6, 2]$.

14.2 Discrete image transforms in image data compression

Image data representation by coefficients of discrete image transforms (see Section 3.2) is the basic idea of this approach. The coefficients are ordered according to their importance, i.e., according to their contribution to the image information content, and the least important (low-contribution) coefficients are omitted. Coefficient importance can be judged, for instance, in correspondence to spatial or gray-level visualization abilities of the display; image correlation can then be avoided and data compression results.

The *Karhunen-Loève* transform is the foremost in removing correlated image data. This transform builds a set of non-correlated variables with decreasing variance. The variance of a variable is a measure of its information content; therefore, a compression strategy is based on considering only transform variables with high variance. More details about the Karhunen-Loève transform can be found in Section 3.2.10.

The Karhunen-Loève transform is computationally expensive, with a two-dimensional transform of an $M \times N$ image having computational complexity $\mathcal{O}(\mathcal{M}^\epsilon \mathcal{N}^\epsilon)$. It is the

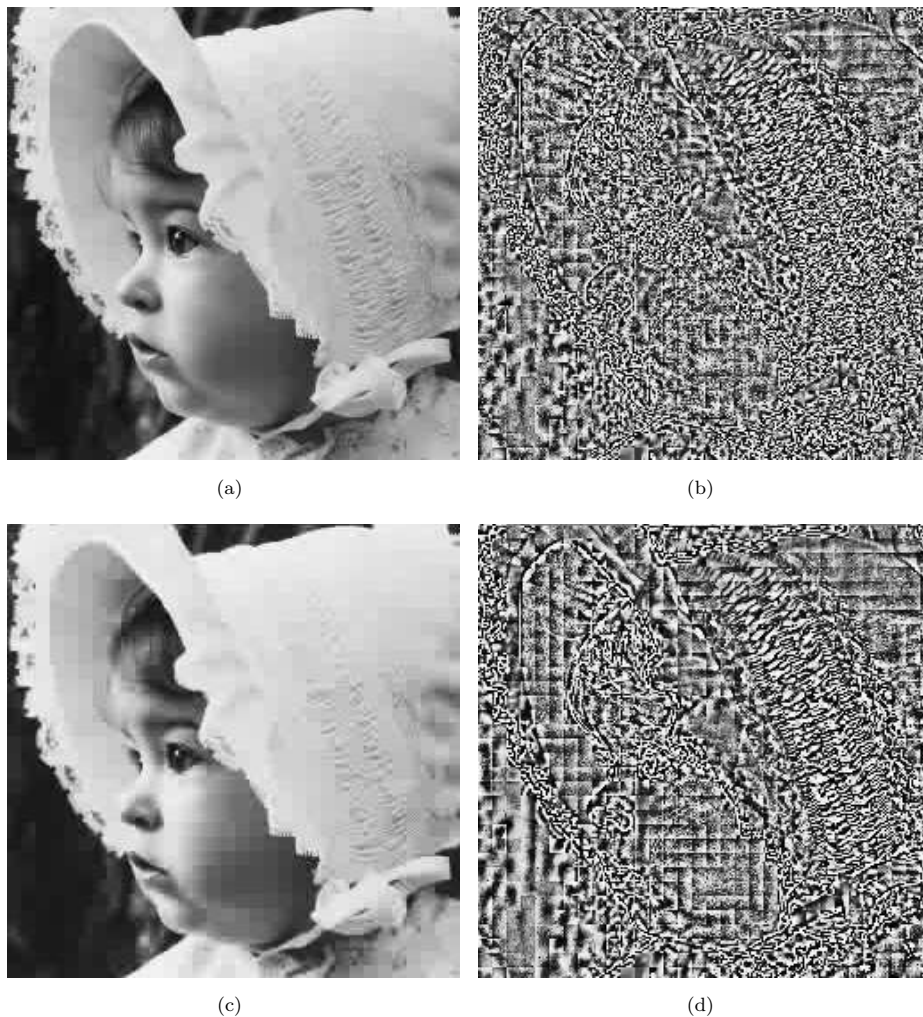


Figure 14.2: Discrete cosine image compression applied to subblocks of 8×8 pixels as in JPEG. (a) Reconstructed image, compression ratio $K = 6.2$. (b) difference image—differences between pixel values in the original and the reconstructed image ($K = 6.2$); the maximum difference is 56 gray-levels, mean squared reconstruction error $\text{MSE} = 32.3$ gray-levels (the image is histogram equalized for visualization purposes). (c) Reconstructed image, compression ratio $K = 10.5$. (d) Difference image—differences between pixel values in the original and the reconstructed image ($K = 10.5$); the maximum difference is 124 gray-levels, $\text{MSE} = 70.5$ gray-levels (the image is histogram equalized for visualization purposes). *Courtesy of A. Kruger, G. Prause, The University of Iowa.*

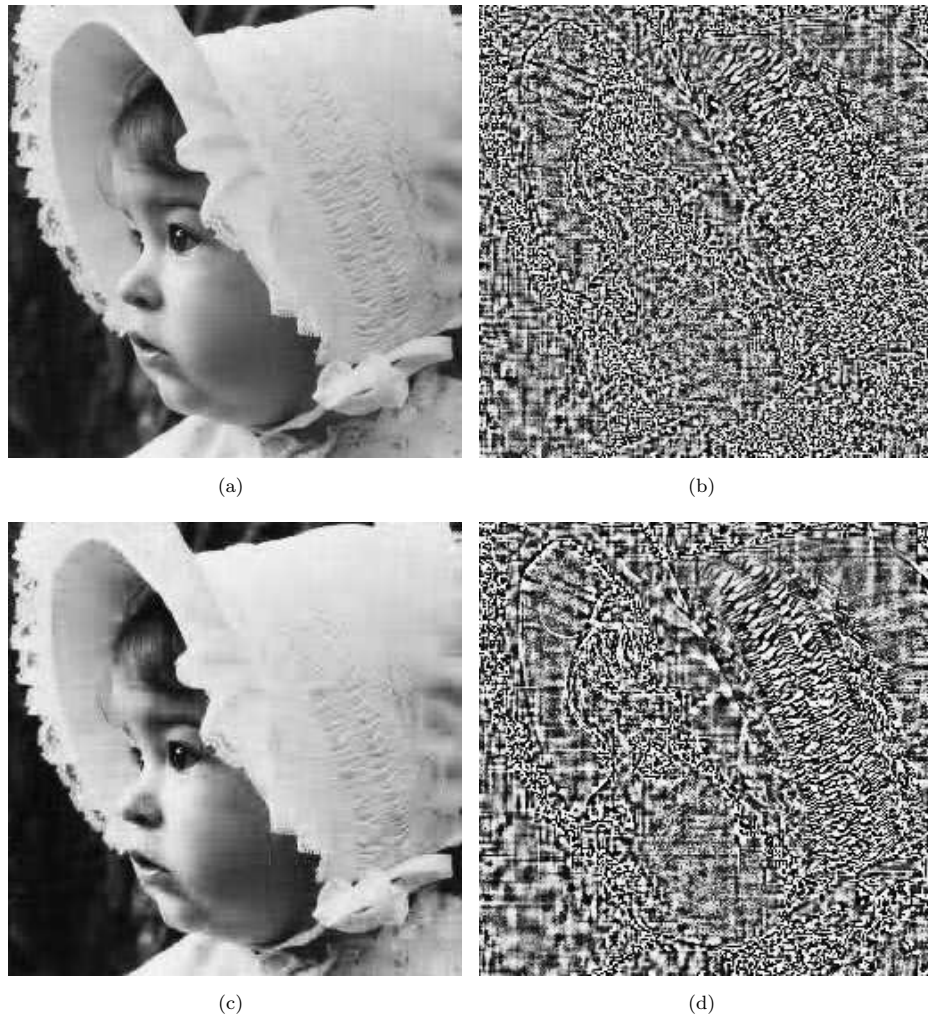


Figure 14.3: Wavelet image compression. (a) Reconstructed image, compression ratio $K = 6.2$. (b) Difference image—differences between pixel values in the original and the reconstructed image ($K = 6.2$); the maximum difference is 37 gray-levels, mean squared reconstruction error MSE = 32.0 gray-levels (the image is histogram equalized for visualization purposes). (c) Reconstructed image, compression ratio $K = 10.5$. (d) Difference image—differences between pixel values in the original and the reconstructed image ($K = 10.5$); the maximum difference is 79 gray-levels, MSE = 65.0 gray-levels (the image is histogram equalized for visualization purposes). *Courtesy of G. Prause and A. Kruger, The University of Iowa.*

only transform that guarantees non-correlated compressed data, and the resulting compression is optimal in the statistical sense. This makes the transform basis vectors image dependent, which also makes the transform difficult to apply for routine image compression. Therefore, it is often only used as a benchmark to evaluate other transforms. For example, one reason for the popularity of the discrete cosine transform DCT-II is that its performance approaches the Karhunen-Loève transform better than others.

Other discrete image transforms (Section 3.2) are computationally less demanding; fast algorithms of these transforms have computational complexity $\mathcal{O}(MN \log_{\epsilon}(MN))$. Cosine, Fourier, Hadamard, Walsh, or binary transforms are all suitable for image data compression. If an image is compressed using discrete transforms, it is usually divided into, e.g., 16×16 pixel, subimages to speed up calculations, and then each subimage is transformed and processed separately. The same is true for image reconstruction, with each subimage being reconstructed and placed into the appropriate image position. This image segmentation into a grid of subimages does not consider any possible data redundancy caused by subimage correlation even if this correlation is the most serious source of redundancy. *Recursive block* coding [Farelle, 1990] is an approach to reducing inter-block redundancy and tiling effects (blockiness). The most popular image transform used for image compression is the discrete cosine transform with many modifications, and variations of wavelet transforms (Section 3.2.7).

Figure 14.2 illustrates the use of the discrete cosine transform. The DCT-II applied here provides good compression with low computational demands, the compression ratios being $K = 6.2$ and $K = 10.5$. The lower compression ratio was achieved after setting 90.0% of the transform coefficients to zero; the higher ratio after setting 94.9% of the coefficients to zero. Note that square blocks resulting from **DCT compression** and reconstruction decrease image quality for larger compression ratios. Consequently, **wavelet image compression** is of interest, since it can be efficiently applied to the entire image and thus the square artifacts are not present. Wavelet compression consists of the same steps as DCT compression, but the DCT is replaced by a wavelet transform followed by generally identical quantization and coding. Figure 14.3 shows the reconstructed image after wavelet compression with two different compression ratios, $K = 6.2$ and $K = 10.5$. The lower compression ratio (Figure 14.3a,b) was achieved after setting 89.4% of the transform coefficients to zero, the higher ratio (Figure 14.3a,b) resulted after setting 94.4% of the coefficients to zero. Note that no blocking artifacts exist.

While DCT compression is the basis for the widely used JPEG compression standard (Section 14.9.1), wavelet compression has become the basis for JPEG-2000 (Section 14.9.2).

14.3 Predictive compression methods

Predictive compressions use image information redundancy (correlation of data) to construct an estimate $\hat{f}(i, j)$ of the gray-level value of an image element (i, j) from values of gray-levels in its neighborhood. In image parts where data are not correlated, the estimate $\hat{f}$ will not match the original value. The differences between estimates and reality, which may be expected to be relatively small in absolute terms, are coded and transmitted (stored) together with prediction model parameters—the whole set now represents compressed image data. The gray value at location (i, j) is reconstructed from a

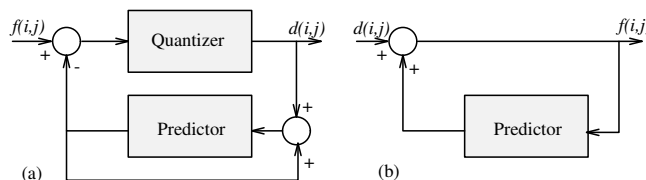


Figure 14.4: Differential pulse code modulation: (a) compression; (b) reconstruction. © Cengage Learning 2015.

computed estimate $\tilde{f}(i, j)$ and the stored difference $d(i, j)$

$$d(i, j) = \tilde{f}(i, j) - f(i, j). \quad (14.6)$$

This method is called **differential pulse code modulation** (DPCM)—its block diagram is presented in Figure 14.4. Experiments show that a linear predictor of the third order is sufficient for estimation in a wide variety of images. If the image is processed line by line, the estimate $\tilde{f}$ can be computed as

$$\tilde{f}(i, j) = a_1 f(i, j - 1) + a_2 f(i - 1, j - 1) + a_3 f(i - 1, j), \quad (14.7)$$

where a_1, a_2, a_3 are image prediction model parameters. These parameters are set to minimize the mean quadratic estimation error e

$$e = \mathcal{E} \left\{ (\tilde{f}(i, j) - f(i, j))^2 \right\}, \quad (14.8)$$

and the solution, assuming f is a stationary random process with a zero mean, using a predictor of the third order, is

$$\begin{aligned} a_1 R(0, 0) + a_2 R(0, 1) + a_3 R(1, 1) &= R(1, 0), \\ a_1 R(0, 1) + a_2 R(0, 0) + a_3 R(1, 0) &= R(1, 1), \\ a_1 R(1, 1) + a_2 R(1, 0) + a_3 R(0, 0) &= R(0, 1), \end{aligned} \quad (14.9)$$

where $R(m, n)$ is the autocorrelation function of the random process f (see Chapter 2). The image data autocorrelation function is usually of exponential form and the variance of differences $d(i, j)$ is usually smaller than the variance of the original values $f(i, j)$, since the differences $d(i, j)$ are not correlated. The (probable) relatively small magnitude of the differences $d(i, j)$ makes data compression possible.

Predictive compression algorithms are described in detail in [Rosenfeld and Kak, 1982; Netravali, 1988]. A second order predictive method with variable code length coding of the differences $d(i, j)$ produced the compressed images shown in Figure 14.5 achieving data compression ratios of $K = 3.8$ and $K = 6.2$. Note that horizontal lines and false contours resulting from the predictive compression and reconstruction decrease the image quality for larger compression ratios.

Many modifications of predictive compression methods can be found in the literature, some of them combining predictive compression with other coding schemes [Daut and Zhao, 1990; Zailu and Taxiao, 1990].

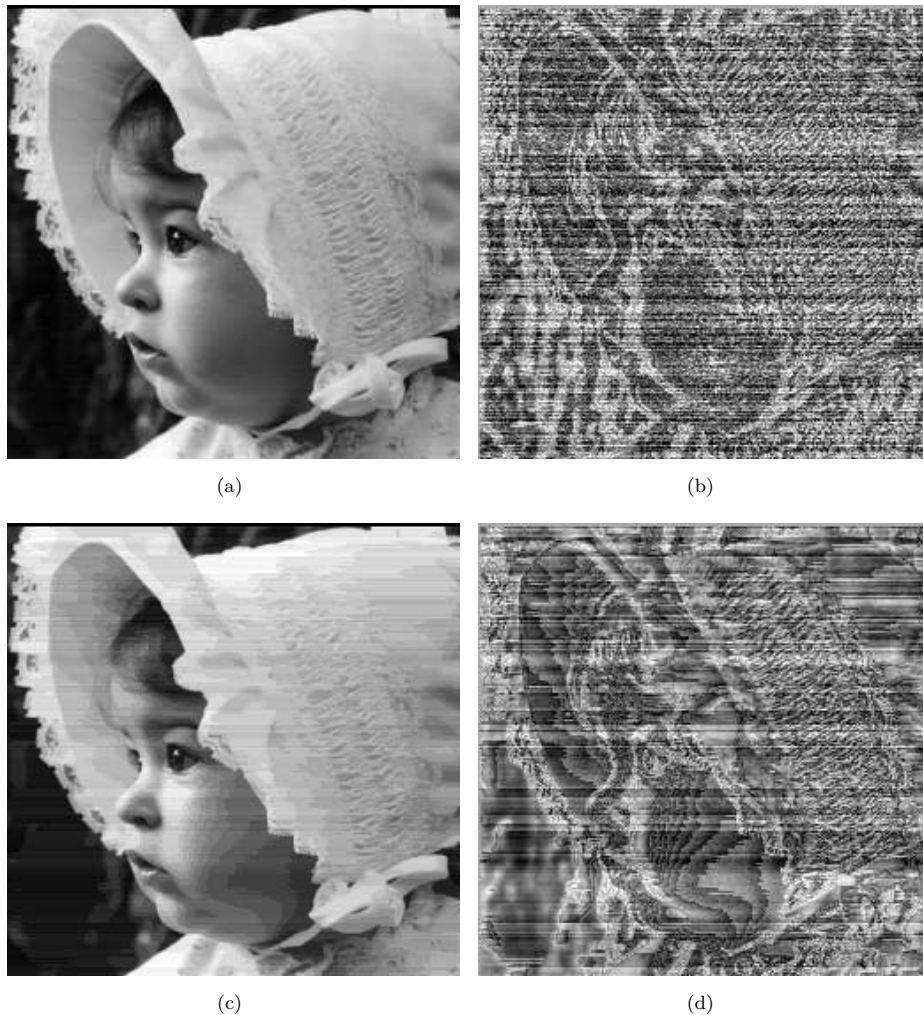


Figure 14.5: Predictive compression. (a) Reconstructed image, compression ratio $K = 3.8$. (b) Difference image—differences between pixel values in the original and the reconstructed image ($K = 3.8$); the maximum difference is 6 gray-levels (the image is histogram equalized for visualization purposes). (c) Reconstructed image, compression ratio $K = 6.2$. (d) Difference image—differences between pixel values in the original and the reconstructed image ($K = 6.2$); the maximum difference is 140 gray-levels (the image is histogram equalized for visualization purposes). *Courtesy of A. Kruger, The University of Iowa.*

14.4 Vector quantization

Dividing an image into small blocks and representing these blocks as vectors is another option [Gray, 1984; Chang et al., 1988; Netravali, 1988; Gersho and Gray, 1992]. The basic idea for this approach comes from information theory (Shannon's rate distortion theory), which states that better compression performance can always be achieved by coding vectors instead of scalars. Input data vectors are coded using unique codewords from a codeword dictionary, and instead of vectors, the vector codes are stored or transmitted. The codeword choice is based on the best similarity between the image block represented by a coded vector and the image blocks represented by codewords from the dictionary. The code dictionary (code book) is transmitted together with the coded data. The advantage of **vector quantization** is a simple receiver structure consisting of a look-up table, but a disadvantage is a complex coder. The coder complexity is not caused directly by the vector quantization principle; the method can be implemented in a reasonably simple way, but the coding will be very slow. To increase the processing speed, data structures such as K-D trees (see Section 9.2.2) and other special treatments are needed which increase the coder complexity. Further, the necessary statistical properties of images are usually not available. Therefore, the compression parameters must be based on an image training set and the appropriate code book may vary from image to image. As a result, images with statistical properties dissimilar from images in the training set may not be well represented by the code vectors in the look-up table. Furthermore, edge degradation may be more severe than with other techniques. To decrease the coder complexity, the coding process may be divided into several levels, two being typical. The coding process is hierarchical, using two or more code books according to the number of coding levels. However, the combination of a complex coder facilitating high compression ratios and a simple decoder may be advantageous in **asymmetric** applications when the image is compressed once and decompressed many times. Within such a scenario, the higher compression ratio gained by the more complex coder and/or more time-consuming compression algorithm does not matter as long as the decompression process is simple and fast. Multimedia encyclopedias and paperless publishing serve as good examples. On the other hand, in **symmetric** applications such as video conferencing, similar complexity of coding and decoding operations is required.

A modification that allows blocks of variable size is described in [Boxerman and Lee, 1990], where a segmentation algorithm is responsible for detecting appropriate image blocks. The block vector quantization approach may also be applied to compression of image sequences. Identifying and processing only blocks of the image that change noticeably between consecutive frames using vector quantization and DPCM is also possible. Hybrid DPCM combined with vector quantization of colored prediction errors is presented in [De Lameillieure and Bruylant, 1990].

14.5 Hierarchical and progressive compression methods

Multi-resolution pyramids may also be used for efficient hierarchical image compression. **Run length** codes (Section 4.2.2, Figure 4.4) identify long runs of the same value pixels, and store them as this value together with a word count. If the image is characterized by such long runs, this will significantly reduce storage requirements. A similar approach may be applied to image pyramids. A substantial reduction in bit volume can be obtained

by merely representing a source as a pyramid [Rao and Pearlman, 1991], and even more significant reduction can be achieved for images with large areas of the same gray-level if a quadtree coding scheme is applied (see Section 4.3.2). An example is given in Figure 14.6, where the principle of quadtree image compression is presented. Large image areas of the same gray-level can be represented in higher-level quadtree nodes without the necessity of including lower-level nodes in the image representation [White, 1987]. Clearly, the compression ratio achieved is image dependent and a fine checkerboard image, for instance, will not be represented efficiently using quadtrees. Modifications of the basic method exist, some of them successfully applied to motion image compression [Strobach, 1990] or incorporating hybrid schemes [Park and Lee, 1991].

This compression approach affords the feasibility of progressive image transmission and the idea of smart compression.

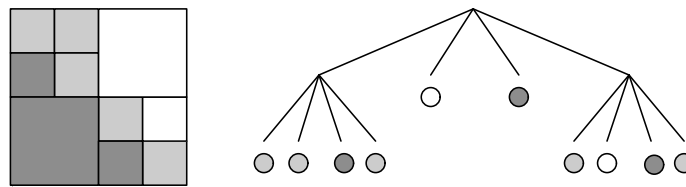


Figure 14.6: Quadtree image compression: original image and corresponding quadtree. © Cengage Learning 2015.

Progressive image transmission is based on the fact that transmitting all image data may not be necessary under some circumstances. Imagine a situation in which an operator is searching an image database looking for a particular image. If the transmission is based on a raster scanning order, all the data must be transmitted to view the whole image, but often it is not necessary to have the highest possible image quality to find the image for which the operator is looking. Images do not have to be displayed with the highest available resolution, and lower resolution may be sufficient to reject an image. This approach is commonly used by WWW image transmissions. In progressive transmission, the images are represented in a pyramid structure, the higher pyramid levels (lower resolution) being transmitted first. The number of pixels representing a lower-resolution image is substantially smaller and thus the user can decide from lower-resolution images whether further image refinement is needed. A standard M-pyramid (mean or matrix pyramid) consists of about one third more nodes than the number of image pixels. Several pyramid encoding schemes have been designed to decrease the necessary number of nodes in pyramid representation: reduced sum pyramids, difference pyramids, and reduced difference pyramids [Wang and Goldberg, 1989]. The reduced difference pyramid has the number of nodes exactly equal to the number of image pixels and can be used for a lossless progressive image transmission with some degree of compression. Using an appropriate interpolation method in the image reconstruction stage, reasonable image quality can be achieved at a bit rate of less than 0.1 bit/pixel and excellent quality at a bit rate of about 1.2 bits/pixel. Progressive image transmission is illustrated in Figure 3.11, where a sequence of four resolutions is presented. Considering a hypothetical progressive image transmission, a $1/8$ -resolution image is transmitted first (Figure 3.11d). Next, the image is transmitted and displayed in $1/4$ resolution (Figure 3.11c), followed by $1/2$ resolution (Figure 3.11b) and then full resolution (Figure 3.11a).

Smart compression is based on the properties of human visual sensors. The spatial resolution of the eye decreases significantly with increasing distance from the optical axis, so the eye only sees in high resolution in a very small area close to the point where it is focused. Similarly, as with image displays, where it does not make sense to display or even transmit an image in higher resolution than that of the display device, it is not necessary to display an image in full resolution in areas where the eyes are not focused. This is the principle of smart image compression. The main difficulty is in determining the areas of interest in the image on which the user will focus. When considering a smart progressive image transmission, the image is first transmitted in higher resolution in areas of interest—this improves the subjective rating of transmission speed as sensed by a user. The areas of interest may be obtained in a feedback control manner by tracking the user's eyes (assuming the communication channel is fast enough). The image point on which the user is focused may be used to increase resolution in that particular area so that the most important data are transmitted first. This smart image transmission and compression may be extremely useful if applied to dynamic image generators in driving or flight simulators, or to high-definition television.

14.6 Comparison of compression methods

The main goal of image compression is to minimize image data volume with no significant loss of information, and all basic image compression groups have advantages and disadvantages. Transform-based methods better preserve subjective image quality, and are less sensitive to statistical image property changes both inside a single image and between images. Prediction methods, on the other hand, can achieve higher compression ratios in a much less expensive way, tend to be much faster than transform-based or vector quantization compression schemes, and can easily be realized in hardware. If compressed images are transmitted, an important property is insensitivity to transmission channel noise. Transform-based techniques are significantly less sensitive to channel noise—if a transform coefficient is corrupted during transmission, the resulting image distortion is spread homogeneously through the image or image part and is not too disturbing. Erroneous transmission of a difference value in prediction compressions causes not only an error in a particular pixel, it influences values in the neighborhood because the predictor involved has a considerable visual effect in a reconstructed image. Vector quantization methods require a complex coder, their parameters are very sensitive to image data, and they blur image edges. The advantage is in a simple decoding scheme consisting of a look-up table only. Pyramid-based techniques have a natural compression ability, and are suitable for dynamic image compression and for progressive and smart transmission approaches.

Hybrid compression methods combine good properties of the various groups. A hybrid compression of three-dimensional image data (two spatial dimensions plus one spectral dimension) is a good example. A two-dimensional discrete transform (cosine, Hadamard, ...) is applied to each mono-spectral image followed by a predictive compression in the third dimension of spectral components. Hybrid methods combine the different dimensionalities of transform compressions and predictive compressions. As a general rule, at least a one-dimensional transform compression precedes predictive compression steps. In addition to combinations of transform and predictive approaches, predictive approaches are often combined with vector quantization.

14.7 Other techniques

Various other compression methods exist. If an image is quantized into a small number of gray-levels and has a small number of regions of the same gray-level, an effective compression method may be based on **coding region borders** [Wilkins and Wintz, 1971]. Image representation by its *low and high frequencies* is another method—reconstruction is a superposition of inverse transforms of low- and high-frequency components. The low-frequency image can be represented by a significantly smaller volume of data than the original, while the high-frequency image has significant image edges only and can be represented efficiently [Graham, 1967]. **Region growing** compression stores an algorithm for region growing from seed points, each region being represented by its seed point. If an image can be represented only by region seed points, significant compression is achieved.

Block truncation coding divides an image into small square blocks of pixels and each pixel value in a block is truncated to one bit by thresholding and moment preserving selection of binary levels [Delp and Mitchell, 1979; Rosenfeld and Kak, 1982; Kruger, 1992]. One bit value per pixel has to be transmitted, together with information describing how to recreate the moment preserving binary levels during reconstruction. This is fast and easy to implement. **Visual pattern image** coding is capable of high-quality compression with very good compression ratios (30:1) and is exceptionally fast [Silsbee et al., 1991].

Fractal image compression is another approach offering extremely high compression ratios and high-quality image reconstruction. Additionally, because fractals are infinitely magnifiable, fractal compression is resolution independent and so a single compressed image can be used efficiently for display in any resolution, including some higher than the original [Furht et al., 1995]. Breaking an image into pieces (fractals) and identifying self-similar ones is the principle of the approach [Barnsley and Hurd, 1993; Fisher, 1994]. First, the image is partitioned into non-overlapping domain regions of any size and shape that completely cover it. Then, larger range regions are defined that can overlap and need not cover the entire image. These range regions are geometrically transformed using affine transforms (Section 5.2.1) to match the domain regions. Then the set of affine coefficients together with information about the selection of domain regions represents the fractal image encoding. The fractally compressed images are stored and transmitted as recursive algorithms—sets of equations with instructions on how to reproduce the image. Clearly, this is compute demanding. However, decompression is simple and fast; domain regions are iteratively replaced with appropriately geometrically transformed range regions using the affine coefficients. Thus, fractal compression is another example of useful asymmetric compression-decompression scheme.

14.8 Coding

In addition to techniques designed explicitly to cope with 2D (or higher-dimensional) data, there is a wide range of well-known algorithms designed with serial data (e.g., simple text files) in mind. Very well known is **Huffman encoding**, which can provide optimal compression and error-free decompression [Rosenfeld and Kak, 1982]. The main idea of Huffman coding is to represent data by codes of variable length, with more frequent data being represented by shorter codes. Many modifications of the original algorithm

[Huffman, 1952] exist, with adaptive Huffman coding algorithms requiring only one pass over the data [Knuth, 1985; Vitter, 1987]. The **Lempel-Ziv** (or Lempel-Ziv-Welch, LZW) algorithm of **dictionary-based** coding [Ziv and Lempel, 1978; Nelson, 1989] has found wide favor as a standard compression algorithm. Here, data are represented by pointers referring to a dictionary of symbols.

These, and a number of similar techniques, are in widespread use for de-facto standard image representations which are popular for WWW image exchange. Of these, the *GIF* format (Graphics Interchange Format) is frequently used. GIF is designed for the encoding of RGB images (and the appropriate palette) with pixel depths between 1 and 8 bits. Blocks of data are encoded using the LZW algorithm. GIF has two main versions—87a and 89a [Compuserve, 1989], the latter supporting the storing of text and graphics in the same file. Additionally, *TIFF* (Tagged Image File Format) is widely encountered. TIFF has gone through a number of versions to incorporate RGB color, compressed color (LZW), other color formats, and ultimately (in Version 6 [Aldus, 1992]), JPEG compression (see Section 14.9)—these versions all have backward compatibility. There are some recorded problems with the JPEG implementation, and TIFF has a reputation for being complex, although this is undeserved and it is a powerful programmer's tool.

14.9 JPEG and MPEG image compression

There is an increasing effort to achieve standardization in image compression. The Joint Photographic Experts Group (JPEG) has developed an international standard for general purpose, color, still image compression. As a logical extension of JPEG still image compression, the Motion Picture Experts Group (MPEG) standard was developed for full-motion video image sequences with applications to digital video distribution and high-definition television (HDTV) in mind.

14.9.1 JPEG—still image compression

The JPEG compression system is widely used in many application areas. Four compression modes are furnished:

- Sequential DCT-based compression.
- Progressive DCT-based compression.
- Sequential lossless predictive compression.
- Hierarchical lossy or lossless compression.

While the lossy compression modes were designed to achieve compression ratios around 15 with very good or excellent image quality, the quality deteriorates for higher compression ratios. A compression ratio between 2 and 3 is typically achieved in the lossless mode.

Sequential JPEG compression

Following Figure 14.1, sequential JPEG compression consists of a forward DCT transform, a quantizer, and an entropy encoder, while decompression starts with entropy decoding followed by dequantizing and inverse DCT.

In the compression stage, the unsigned image values from the interval $[0, 2^b - 1]$ are first shifted to cover the interval $[-2^{b-1}, 2^{b-1} - 1]$. The image is then divided into 8×8 blocks and each block is independently transformed into the frequency domain using the DCT-II transform (Section 3.2.6, equation 3.46). Many of the 64 DCT coefficients have zero or near-zero values in typical 8×8 blocks, which forms the basis for compression. The 64 coefficients are quantized using a quantization table $Q(u, v)$ of integers from 1 to 255 that is specified by the application to reduce the storage/transmission requirements of coefficients that contribute little or nothing to the image content. The following formula is used for quantization:

$$F_Q(u, v) = \text{round} \left(\frac{F(u, v)}{Q(u, v)} \right). \quad (14.10)$$

After quantization, the DC coefficient $F(0, 0)$ is followed by the 63 AC coefficients that are ordered in a 2D matrix in a zigzag fashion according to increasing frequency. The DC coefficients are then encoded using predictive coding (Section 14.3), the rationale being that average gray-levels of adjacent 8×8 blocks (DC coefficients) tend to be similar.

The last step of the sequential JPEG compression algorithm is entropy encoding. Two approaches are specified by the JPEG standard. The baseline system uses simple Huffman coding, while the extended system uses arithmetic coding and is suitable for a wider range of applications.

Sequential JPEG decompression uses all the steps described above in the reverse order. After entropy decoding (Huffman or arithmetic), the symbols are converted into DCT coefficients and dequantized:

$$F'_Q(u, v) = F_Q(u, v) Q(u, v), \quad (14.11)$$

where again, the $Q(u, v)$ are quantization coefficients from the quantization table that is transmitted together with the image data. Finally, the inverse DCT transform is performed according to equation (3.47) and the image gray values are shifted back to the interval $[0, 2^b - 1]$.

The JPEG compression algorithm can be extended to color or multi-spectral images with up to 256 spectral bands.

Progressive JPEG compression

The JPEG standard also facilitates progressive image transmission (Section 14.5). In the progressive mode, a sequence of scans is produced, each containing a coded subset of DCT coefficients. Thus, a buffer is needed at the output of the quantizer to store all DCT coefficients of the entire image. These coefficients are selectively encoded.

Three algorithms are defined as part of the JPEG progressive compression standard: **progressive spectral selection**, **progressive successive approximation**, and the **combined progressive algorithm**. In the progressive spectral selection approach, the DC coefficients are transmitted first, followed by groups of low-frequency and higher-frequency coefficients. In the progressive successive approximation, all DCT coefficients are sent first with lower precision, and their precision is increased as additional scans are transmitted. The combined progressive algorithm uses both of the above principles together.

Sequential lossless JPEG compression

The lossless mode of the JPEG compression uses a simple predictive compression algorithm and Huffman coding to encode the prediction differences (Section 14.3).

Hierarchical JPEG compression

Using the hierarchical JPEG mode, decoded images can be displayed either progressively or at different resolutions. An image pyramid is created and each lower-resolution image is used as a prediction for the next-higher level (Section 14.5). Sequential DCT, progressive DCT, or lossless modes can be used to encode the lower-resolution images.

In addition to still image JPEG compression, motion JPEG (MJPEG) compression exists that can be applied to real-time full motion applications. However, MPEG compression represents a more common standard and is described below.

14.9.2 JPEG–2000 compression

JPEG–2000 is an international standard for still image compression which overcomes some limitations of the original standard (Section 14.9.1). Despite the naming similarity, it is not an extension of JPEG—rather, it is a different, very powerful and flexible environment for image compression. Its flexibility allows compression of different types of still images (bi-level, gray-level, color, multi-band) with different characteristics (natural images, scientific, medical, military imagery, text, rendered graphics) within a unified system. It removes the need for different compression mechanisms for lossless and lossy compression by representing the lossless compression as a cohesive extension of lossy compression. This important paradigm shift compared to the JPEG standard permits the compression of image data in a lossless manner and—at a later time—a selective data removal to represent images in a lossy fashion while increasing the compression ratio. Note that this lossless and lossy behavior can be achieved from the same compressed image data source—this feature is called **quality scalability**. Another JPEG–2000 feature is the option of **resolution scalability**, which allows the extraction of lower resolution images from the same data source. Additionally, **spatial scalability** provides a tool to selectively reconstruct individually defined regions from the compressed image data source.

While the standard creates a unified image compression environment, it only specifies the decoder operations, bitstream syntax, and file format, effectively allowing for future improvements and innovations of the coding operation. For encoding, two primary paths and several options exist. A reversible component transform (RCT) is used with the 5×3 wavelet filter (Section 3.2.7) whenever lossless compression is desired. Decreased bit rates and increased compression ratios can be achieved by truncation during the quantization step—of course yielding a decrease in image quality.

For purely lossy coding, the YCbCr transform (shown here for RGB image data) transforms the RGB signal to the intensity component Y and two color components (C_b for blue and C_r for red) as follows:

$$\begin{aligned} Y &= +0.299 R + 0.587 G + 0.114 B, \\ C_b &= -0.168736 R - 0.331264 G + 0.5 B, \\ C_r &= +0.5 R - 0.418688 G - 0.081312 B. \end{aligned} \tag{14.12}$$

After this, a 9×7 wavelet transform, and arbitrary quantization by division, is used in addition to truncation. Both of these main paths have several options for identification of the region of interest, coding options to trade complexity and performance, and choices about the amount of scalability in the bitstream.

For better understanding, the high-level coverage of the JPEG-2000 compression approach given below exploits the compression rather than the reconstruction path. More detailed information can be found in [Taubman and Marcellin, 2001; Colyer and Clark, 2003]. Figure 14.7 gives an overview of the JPEG-2000 compression approach; the main data path is shown in the bottom of the schematic diagram.

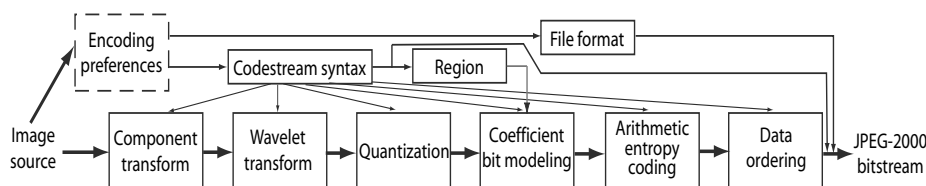


Figure 14.7: Schematic of JPEG-2000 data compression. © Cengage Learning 2015.

To start the compression, the image is divided into rectangular, non-overlapping tiles on a regular grid (border tiles may be sized as needed). Arbitrary tile sizes are allowed, up to using a single tile representing the entire image. The *component transform* block takes the original image data as input and decorrelates the image components of a multi-band image—typically the R,G,B channels of the color image. Decorrelation yields an improved compression performance and allows for visually relevant quantization. As described above, when using the lossless path, RCT is employed which maps integers to integers. When the lossy (irreversible) path is used the floating-point YCbCr transform is employed in the same way as it is used in the original color JPEG compression.

The *wavelet transform* is the heart of the JPEG-2000 compression and can be performed in two ways. Both of the wavelet transform options provide lower resolution and spatial decorrelation of the images. The 9×7 biorthogonal Daubechies filter provides the highest compression, while the Le Gall 5×3 filter is of a lower complexity and allows lossless compression. The advanced parts of the JPEG-2000 standard facilitate simultaneous use of multiple wavelets including user-defined wavelet transforms for which the coefficients are specified in the bitstream. One of the most obvious artifacts of the original JPEG compression is the blocky character of the reconstructed images due to the rectangular image blocks to which the DCT is applied. Since the wavelet compression can be applied to the entire image, which is converted into a series of wavelets, the blockiness may be completely removed. Even if the block-based wavelet transformation is employed, the blockiness is substantially decreased; smoother color toning and clearer edges result in locations of sharp changes of color.

The *quantization* step offers a trade-off between compression ratio and image quality. Similar to JPEG, the wavelet coefficients can be divided by a different value for each image subband. Some of the coded data can be discarded to increase the compression ratio. Many approaches exist this [Zeng et al., 2000]. The *context model* groups quantized wavelet coefficients according to their statistical similarity to increase the efficiency of their compression. Individual coefficient bitplanes are processed by one of three coding passes described in [Taubman et al., 2000]. The *binary arithmetic coder* provides lossless

compression of each coding pass of quantized wavelet coefficients. Portions of the data output from the arithmetic coder form the compressed data *bitstream*. The coded data are collected into *packets*. Each packet has a compressed header allowing data to be accessed in almost any order, although some order requirements are maintained. Specific orderings facilitate several progressive compression options, e.g., according to the resulting image resolution, quality, location, or their combinations.

The *codestream syntax* prescribes marker segments, which determine the location of the coded data with respect to a given spatial image location, resolution, and quality. Any image-related data which are not directly needed to reconstruct the image are stored in the optional *file format* data. Data format information was included to avoid formation of non-standard proprietary formats as happened with the original JPEG standard. The file format begins with a unique signature, has a profile indicator, and repeats the width, height, and depth information from the codestream. The file format data may also contain a limited color specification, capture and display resolution, intellectual property rights information, and/or additional meta-data [Boliak et al., 2000].

The benefits of JPEG-2000 are wide [Gormish et al., 2000]. In web-based applications, it allows initial and quick display of a low resolution version of the image, say of a map. Later, any part of the map can be requested via the region of interest selection and the server would only provide the necessary additional data for that spatial region at the required resolution. Further, if the user requests a printout of that region, a higher resolution version that is matched to the printer resolution would be fetched over the network and, based on the gray-level or color printer capabilities, only grayscale or color information would be transferred. The selective transmission of only necessary data by the specific application is an inherent and intriguing feature of the JPEG-2000 standard.

Similarly, consider the common situation of storing high resolution digital photographs, and running out of storage space. With current compression approaches, one image must be deleted before we can store another. If photos are stored using JPEG-2000, it is possible incrementally decrease the quality of all stored images in order to make space for new ones.

JPEG-2000 is a much better compression tool than JPEG when high image quality is demanded, even when using lossy compression. For lossy compression, data has shown that JPEG-2000 can typically compress images 20–200% more than JPEG. Note that JPEG-2000 can handle up to 256 image channels while the original JPEG was, due to its common implementation, limited to only 3-channel color data. JPEG-2000 compression ratios of about 2.5 are typical for lossless compression. *Motion JPEG* (discussed in the previous section) is frequently used for editing production-quality video without the existence of an internationally accepted standard. JPEG-2000 includes standardized *Motion JPEG-2000* format.

For applications requiring either higher image quality or low bitrates, and if previously unavailable and newly included features are of interest, JPEG-2000 is the compression standard of choice. The original JPEG standard is not likely to disappear quickly. It will probably survive as a useful tool for lower complexity applications.

14.9.3 MPEG—full-motion video compression

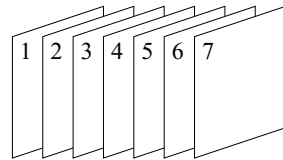
Video and associated audio data can be compressed using *MPEG*. Using inter-frame compression, compression ratios of 200 can be achieved in full-motion, motion-intensive video applications maintaining reasonable quality. MPEG facilitates many features of

the compressed video; random access, fast forward/reverse searches, reverse playback, audio-visual synchronization, robustness to error, editability, format flexibility, and cost trade-off [LeGall, 1991; Steinmetz, 1994]. Three standards are frequently cited:

- MPEG-1 for compression of low-resolution (320×240) full-motion video at rates of 1–1.5 Mb/s
- MPEG-2 for higher-resolution standards such as TV and HDTV at rates of 2–80 Mb/s
- MPEG-4 for small-frame full motion compression with slow refresh needs, rates of 9–40 kb/s for video telephony and interactive multimedia such as video conferencing

MPEG can be equally well used for both symmetric and asymmetric applications. Here, video compression will be described; a description of the audio compression that is also part of the standard can be found elsewhere [Pennebaker and Mitchell, 1993; Steinmetz, 1994].

The video data consist of a sequence of image frames. In the MPEG compression scheme, three frame types are defined: *intraframes I*; *predicted frames P*; and *forward, backward, or bi-directionally predicted or interpolated frames B*. Each frame type is coded using a different algorithm; Figure 14.8 shows how the frame types may be positioned in the sequence.



I B B P B B I

Figure 14.8: MPEG image frames. © Cengage Learning 2015.

I-frames are self-contained and coded using a DCT-based compression method similar to JPEG. Thus, I-frames serve as random access frames in MPEG frame streams and consequently are compressed with the lowest compression ratios. P-frames are coded using forward predictive coding with reference to the previous I- or P-frame, and their compression ratio is substantially higher than that for I-frames. B-frames are coded using forward, backward, or bi-directional motion-compensated prediction or interpolation using two reference frames, closest past and future I- or P-frames, and offer the highest compression ratios.

Note that in the hypothetical MPEG stream shown in Figure 14.8, the frames must be transmitted in the following sequence (subscripts denote frame numbers): I_1 – P_4 – B_2 – B_3 – I_7 – B_5 – B_6 – etc.; the frames B_2 and B_3 must be transmitted after frame P_4 to enable frame interpolation used for B-frame decompression. Clearly, the highest compression ratios can be achieved by incorporation of a large number of B-frames; if only I-frames are used, MJPEG compression results. The following sequence seems to be effective for a large number of applications [Steinmetz, 1994]

$$(I B B P B B P B B)(I B B P B B P B B) \dots \quad (14.13)$$

While coding the I-frames is straightforward, coding of P- and B-frames incorporates motion estimation (see also Chapter 16). For every 16×16 block of P- or B-frames, one motion vector is determined for P- and forward or backward predicted B-frames, two motion vectors are calculated for interpolated B-frames. The motion estimation technique is not specified in the MPEG standard, but block matching techniques are widely used, generally following the matching approaches presented in Section 6.4, equation (6.32) [Furht et al., 1995]. After the motion vectors are estimated, differences between the predicted and actual blocks are determined and represent the error terms which are encoded using DCT. As usually, entropy encoding is employed as the final step.

A specific MPEG-4 variant often referred to as *MP4* (MPEG-4 Part 14) is in widespread use, especially for streaming across the Internet. MP4 is a general container that is able to store video, audio, subtitles, stills etc.

14.10 Summary

• Image data compression

- The main goal of image compression is to minimize image data volume with no significant loss of information.
- Image compression algorithms aim to remove redundancy present in data (correlation of data) in a way which makes image reconstruction possible; this is *information preserving compression*.
- A typical image *compression/decompression* sequence consists of data redundancy reduction, coding, transmission, decoding, and reconstruction.
- Data compression methods can be divided into two principal groups:
 - * *Information preserving—lossless*—compressions permit error-free data reconstruction.
 - * *Lossy* compression methods do not preserve information completely.

• Image data properties

- Information content of an image is an important property of which *entropy* is a measure.
- Knowing image entropy, information *redundancy* can be determined.

• Discrete image transforms in image data compression

- Image data are represented by *coefficients* of discrete image transforms. Transform coefficients are *ordered* according to their importance, i.e., according to their contribution to the image information contents. Low-contribution coefficients are omitted.
- To remove correlated (redundant) image data, the *Karhunen-Loève* transform is the most effective.
- *Cosine*, *Fourier*, *Hadamard*, *Walsh*, or *binary* transforms are all suitable for image data compression.

- Performance of the *discrete cosine transform DCT-II* approaches that of the Karhunen-Loève transform better than others. The DCT is usually applied to small image blocks (typically 8×8 pixels), yielding quality-decreasing blocking artifacts for larger compression ratios.
 - Consequently, *wavelet image compression* is of interest because it does not generate square image compression artifacts.
- **Predictive compression methods**
 - Predictive compressions use image information redundancy to construct an estimate of gray-level values from values of gray-levels in the neighborhood.
 - *Differences* between estimates and reality, which are expected to be relatively small in absolute terms, are coded and transmitted together with prediction model parameters.
- **Vector quantization**
 - Vector quantization compression is based on dividing an image into small blocks and representing these blocks as *vectors*.
 - Input data vectors are coded using unique *codewords* from a *codeword dictionary*; instead of vectors, the vector codes are stored or transmitted.
 - The code dictionary (code book) is transmitted together with the coded data.
- **Hierarchical and progressive compression methods**
 - Substantial reduction in bit volume can be obtained by representing a source as a pyramid. More significant reduction can be achieved for images with large areas of the same gray-level in a quadtree coding scheme.
 - Hierarchical compression facilitates *progressive* and *smart* image transmission.
 - Progressive image transmission is based on the fact that transmitting all image data may not be necessary under some circumstances.
 - Smart compression is based on the properties of human visual sensors—it is not necessary to display an image in full resolution in areas where the user's eyes are not focused.
- **Comparison of compression methods**
 - Transform-based methods better preserve subjective image quality, and are less sensitive to statistical image property changes both inside a single image and between images.
 - Prediction methods can achieve larger compression ratios in a less expensive way, and tend to be faster than transform-based or vector quantization compression schemes.
 - Vector quantization methods require a complex coder, their parameters are very sensitive to image data, and they blur image edges. The advantage is in a simple decoding scheme consisting of a look-up table only.

- **Other techniques**

- *Fractal image compression* offers extremely high compression ratios and high-quality image reconstruction. Breaking an image into pieces (fractals) and identifying self-similar ones is the main principle. Fractals are infinitely magnifiable, thus fractal compression is *resolution independent* and a single compressed image can be efficiently used for display in any image resolution.

- **Coding**

- *Huffman encoding* can provide optimal compression and error-free decompression. The main idea of Huffman coding is to represent data by codes of variable length, with more frequent data being represented by shorter codes.

- **JPEG and MPEG image compression**

- JPEG and JPEG-2000 represent international standards in image compression.
- JPEG image compression was developed for general-purpose, color, still image compression. Four JPEG compression modes exist:
 - * Sequential DCT-based compression
 - * Progressive DCT-based compression
 - * Sequential lossless predictive compression
 - * Hierarchical lossy or lossless compression
- JPEG-2000 is designed to overcome some JPEG limitations. Despite the naming similarity, it is different to JPEG.
- JPEG-2000 is wavelet-transform based and offers a rich and flexible set of new functionalities in respect of quality, resolution, and spatial scalability.
- JPEG-2000 typically outperforms JPEG compression in applications requiring either high quality image reconstruction or low bitrate compression.
- The *MPEG standard* was developed for full-motion video image sequences.
- Three standards are frequently cited:
 - * MPEG-1 for compression of low-resolution full-motion video
 - * MPEG-2 for higher-resolution standards
 - * MPEG-4 for small-frame full-motion compression with slow refresh needs

14.11 Exercises

Short-answer questions

S14.1 Explain the difference between lossy and lossless image compression,

S14.2 Give definitions of:

- Image entropy
- Image redundancy
- Compression ratio

- S14.3 How can image entropy be used to determine information redundancy?
- S14.4 Draw a diagram of image compression/transmission/decompression – include all main steps.
- S14.5 Explain the basic idea of image compression using discrete image transforms.
- S14.6 Explain the basic idea of predictive image compression.
- S14.7 Explain the basic idea of image compression based on vector quantization.
- S14.8 What are symmetric and asymmetric image compression applications?
- S14.9 Explain the concept of progressive image transmission and smart compression.
- S14.10 Explain the basic idea of fractal image compression.
- S14.11 What is the basic idea of Huffman coding?
- S14.12 What kind of image data can be compressed using the JPEG and MPEG compression algorithms?
- S14.13 What are the four compression modes available in the JPEG compression standard? Describe the algorithm employed in each of the four modes.
- S14.14 What are the main differences between original JPEG and JPEG-2000 compression schemes?
- S14.15 What image data are the three most frequently used MPEG standards applied to?
- S14.16 Explain the roles of the **I**-, **P**-, **B**- frames in MPEG compression.
- S14.17 MPEG compression – why are individual frames in an image sequence compressed using different approaches?

Problems

- P14.1 Assume a gray-level image represented using 2 bits per pixel. The probabilities of the four gray-levels $\{0, 1, 2, 3\}$ are as follows: $P(0) = 0.1$, $P(1) = 0.3$, $P(2) = 0.5$, $P(3) = 0.1$.
- Determine the image entropy.
 - Determine the information redundancy.
 - Determine the theoretically achievable lossless image compression ratio.
 - What is the minimum size of a 4-level 512×512 image after such lossless compression?
- P14.2 Repeat Problem P14.1 considering a 4-level image in which all gray-levels are equally probable.
- P14.3 Compare entropies of two 4-level images A and B ;
- A : $P(0) = 0.2$, $P(1) = 0.0$, $P(2) = 0.0$, $P(3) = 0.8$
 - B : $P(0) = 0.0$, $P(1) = 0.8$, $P(2) = 0.0$, $P(3) = 0.2$.
- P14.4 Develop a program to compute an estimate of the achievable compression ratio of an image from its gray-level histogram.
- P14.5 Compare image compression performance for several widely available algorithms. Using any image viewer capable of converting between different formats, compare the storage requirements of a variety of image formats such as PBM, GIF, TIFF–no compression, TIFF–LZW compression, JPEG at 50%, 25%, 15%, 5%. By re-opening the stored images, compare the image quality visually; do you see any blocking artifacts if JPEG compression was employed? Order the image formats/compression parameters according to their storage requirements. What can be concluded if you assess the image quality considering the storage requirements?

- P14.6 Use your computer screen-grabbing tools to grab a portion of a text from a window. Store it in a gray-scale mode using GIF and JPEG (75%, 25%, and 5%). Compare the storage requirements and the resulting image quality.
- P14.7 Develop a program for transform-based image compression by applying the Fourier transform to 8×8 blocks (use any 2D discrete Fourier transform implementation—they are easily downloadable). Compute a difference image between the reconstructed image and the original image. Determine the compression ratio and the maximum gray-level difference. Perform the compression in the following ways:
- Retain 15% of the transform coefficients.
 - Retain 25% of the transform coefficients.
 - Retain 50% of the transform coefficients.
- P14.8 Develop a program for DPCM image compression using a third-order predictor.
- P14.9 GIF (lossless) and JPEG (lossy) compressed images are commonly used on the World Wide Web. Determine compression ratios of several GIF and JPEG compressed images; assume that original color images were represented using 24 bits per pixel and gray-scale images by 8 bits per pixel. Which compression scheme seems to provide consistently higher compression ratio? Can you determine by inspection which of the two compression approaches was used?
- P14.10 MPEG compressed image sequences are commonly used on the World Wide Web. Determine compression ratios of several MPEG compressed sequences; assume that original color images were represented using 24 bits per pixel. (An MPEG viewer that provides information about the number of frames in the sequence will be needed.)
- P14.11 Make yourself familiar with solved problems and **Matlab** implementations of selected algorithms provided in the corresponding chapter of the **Matlab Companion** to this text [Svoboda et al., 2008]. The **Matlab Companion** homepage <http://visionbook.felk.cvut.cz> offers images used in the problems, and well-commented **Matlab** code is provided for educational purposes.
- P14.12 Use the **Matlab Companion** [Svoboda et al., 2008] to develop solutions to additional exercises and practical problems provided there. Implement your solutions using **Matlab** or other suitable programming languages.

14.12 References

- Aldus. *TIFF Developer's Toolkit, Revision 6.0*. Aldus Corporation, Seattle, WA, 1992.
- Barnsley M. and Hurd L. *Fractal Image Compression*. A K Peters Ltd., Wellesley, MA, 1993.
- Boliek M., Houchin S., and Wu G. JPEG 2000 next generation image compression system: Features and syntax. In *Proceedings ICIP-2000, Vol. 2*, pages 45–48. IEEE, 2000.
- Boxerman J. L. and Lee H. J. Variable block-sized vector quantization of grayscale images with unconstrained tiling. In *Visual Communications and Image Processing '90*, Lausanne, Switzerland, pages 847–858, Bellingham, WA, 1990. SPIE.
- Chang C. Y., Kwok R., and Curlander J. C. Spatial compression of Seasat SAR images. *IEEE Transactions on Geoscience and Remote Sensing*, 26(5):673–685, 1988.
- Colyer G. and Clark R. *Guide to the practical implementation of JPEG 2000 – PD 6777*. British Standards Institute, London, 2003.
- Compuserve. *Graphics Interchange Format: Version 89a*. CompuServe Incorporated, Columbus, OH, 1989.

- Daut D. G. and Zhao D. Improved DPCM algorithm for image data compression. In *Image Processing Algorithms and Techniques*, Santa Clara, CA, pages 199–210, Bellingham, WA, 1990. SPIE.
- De Lameillieure J. and Bruyland I. Single stage 280 Mbps coding of HDTV using HDPCM with a vector quantizer based on masking functions. *Signal Processing: Image Communication*, 2(3):279–289, 1990.
- Delp E. J. and Mitchell O. R. Image truncation using block truncation coding. *IEEE Transactions on Communications*, 27:1335–1342, 1979.
- Farelle P. M. *Recursive Block Coding for Image Data Compression*. Springer Verlag, New York, 1990.
- Fisher Y. *Fractal Compression: Theory and Applications to Digital Images*. Springer Verlag, Berlin, New York, 1994.
- Furht B., Smoliar S. W., and Zhang H. *Video and Image Processing in Multimedia Systems*. Kluwer, Boston–Dordrecht–London, 1995.
- Gersho A. and Gray R. M. *Vector Quantization and Signal Compression*. Kluwer, Norwell, MA, 1992.
- Gormish M. J., Lee D., and Marcellin M. W. JPEG 2000: Overview, architecture, and applications. In *Proceedings ICIP-2000, Vol. 2*, pages 29–32, 2000.
- Graham D. N. Image transmission by two-dimensional contour coding. *Proceedings IEEE*, 55:336–346, 1967.
- Gray R. M. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984.
- Huffman D. A. A method for the construction of minimum-redundancy codes. *Proceedings of IRE*, 40(9):1098–1101, 1952.
- Knuth D. E. Dynamic Huffman coding. *Journal of Algorithms*, 6:163–180, 1985.
- Kruger A. Block truncation compression. *Dr Dobb's J Software Tools*, 17(4):48–55, 1992.
- LeGall D. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34:45–68, 1991.
- Moik J. G. *Digital Processing of Remotely Sensed Images*. NASA SP-431, Washington, DC, 1980.
- Nelson M. R. LZW data compression. *Dr Dobb's J Software Tools*, 14, 1989.
- Netravali A. N. *Digital Pictures: Representation and Compression*. Plenum Press, New York, 1988.
- Park S. H. and Lee S. U. Pyramid image coder using classified transform vector quantization. *Signal Processing*, 22(1):25–42, 1991.
- Pennebaker W. B. and Mitchell J. L. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, 1993.
- Pratt W. K. *Digital Image Processing*. Wiley, New York, 2nd edition, 1991.
- Rao R. P. and Pearlman W. A. On entropy of pyramid structures. *IEEE Transactions on Information Theory*, 37(2):407–413, 1991.
- Rosenfeld A. and Kak A. C. *Digital Picture Processing*. Academic Press, New York, 2nd edition, 1982.
- Salomon D. *Data Compression: The Complete Reference*. Springer, 2 edition, 2000.
- Sayood K. *Introduction to data compression*. Morgan Kaufmann Publishers, Burlington, USA, 2 edition, 2000.

- Shukla K. K. and Prasad M. *Lossy Image Compression*. Springer, 2011.
- Silsbee P., Bovik A. C., and Chen D. Visual pattern image sequencing coding. In *Visual Communications and Image Processing '90*, Lausanne, Switzerland, pages 532–543, Bellingham, WA, 1991. SPIE.
- Steinmetz R. Data compression in multimedia computing—standards and systems, parts i and ii. *Journal of Multimedia Systems*, 1:166–172 and 187–204, 1994.
- Strobach P. Tree-structured scene adaptive coder. *IEEE Transactions on Communications*, 38(4):477–486, 1990.
- Svoboda T., Kybic J., and Hlavac V. *Image Processing, Analysis, and Machine Vision: A MATLAB Companion*. Thomson Engineering, 2008.
- Taubman D. S., Ordentlich E., Weinberger M., Seroussi G., Ueno I., and Ono F. Embedded block coding in JPEG 2000. In *Proceedings ICIP-2000, Vol. 2*, pages 33–36. IEEE, 2000.
- Taubman D. S. and Marcellin M. W. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer, Boston, MA, 2001.
- Vitter J. S. Design and analysis of dynamic Huffman codes. *Journal of the ACM*, 34(4):825–845, 1987.
- Wang L. and Goldberg M. Reduced-difference pyramid: A data structure for progressive image transmission. *Optical Engineering*, 28(7):708–716, 1989.
- White R. G. Compressing image data with quadrees. *Dr Dobbs's J Software Tools*, 12(3):16–45, 1987.
- Wilkins L. C. and Wintz P. A. Bibliography on data compression, picture properties and picture coding. *IEEE Transactions on Information Theory*, 17:180–199, 1971.
- Zailu H. and Taxiao W. MDPCM picture coding. In *1990 IEEE International Symposium on Circuits and Systems*, New Orleans, LA, pages 3253–3255, Piscataway, NJ, 1990. IEEE.
- Zeng W., Daly S., and Lei S. Visual optimization tools in JPEG 2000. In *Proceedings ICIP-2000, Vol. 2*, pages 37–40. IEEE, 2000.
- Ziv J. and Lempel A. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

Chapter 15

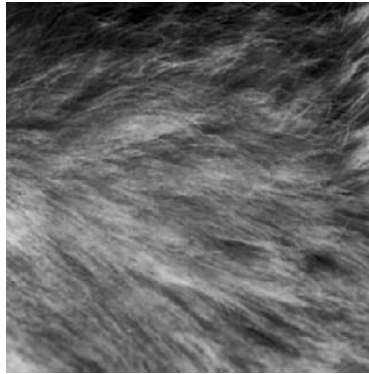
Texture

Texture refers to properties that represent the surface or structure of an object (in reflective or transmissive images, respectively); it is widely used, and perhaps intuitively obvious, but has no precise definition due to its wide variability. We might define texture as *something consisting of mutually related elements*; therefore we are considering a group of pixels and the texture described is highly dependent on the number considered [Haralick, 1979]. Examples are shown in Figure 15.1; dog fur, grass, river pebbles, cork, checkered textile, and knitted fabric.

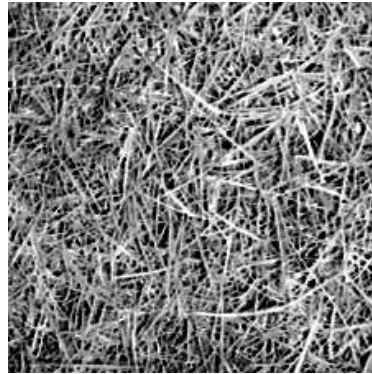
Texture consists of texture **primitives** or texture **elements**, sometimes called **texels**. Primitives in grass and dog fur are represented by several pixels and correspond to a stalk or a pile; cork is built from primitives that are comparable in size with pixels. It is difficult, however, to define primitives for the checkered textile or fabric, which can be defined by at least two hierarchical levels. The first corresponds to textile checks or knitted stripes, and the second to the finer texture of the fabric or individual stitches. As we have seen in many other areas, this is a problem of **scale**; texture description is **scale dependent**.

The main aim of texture analysis is recognition and texture-based shape analysis. Textured properties of regions were referred to many times while considering image segmentation (Chapter 6), and derivation of shape from texture was discussed in Chapter 11. Texture is usually described as **fine**, **coarse**, **grained**, **smooth**, etc., implying that some more precise features must be defined to make machine recognition possible. Such features can be found in the texture **tone** and **structure** [Haralick, 1979]. Tone is based mostly on pixel intensity properties in the primitive, while structure is the spatial relationship between them.

Each pixel can be characterized by its location and tonal properties. A texture primitive is a contiguous set of pixels with some tonal and/or regional property, and can be described by its average intensity, maximum or minimum intensity, size, shape, etc. The spatial relationship of primitives can be random, or they may be pairwise dependent, or some number of primitives can be mutually dependent. Image texture is then described by the number and types of primitives and by their spatial relationship.



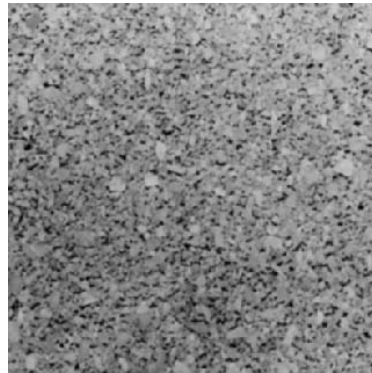
(a)



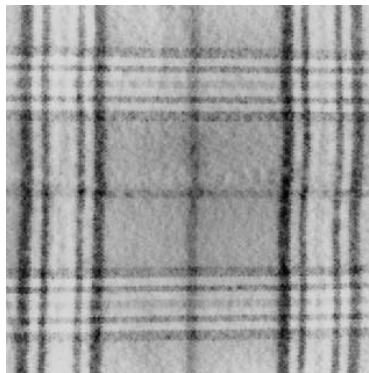
(b)



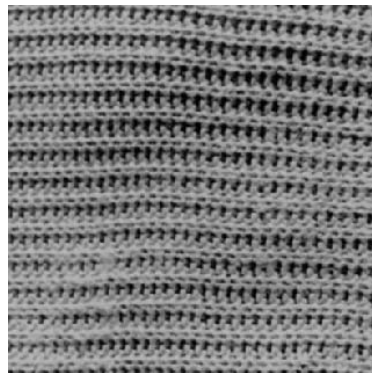
(c)



(d)



(e)



(f)

Figure 15.1: Textures. (a) Dog fur. (b) Grass. (c) River pebbles. (d) Cork. (e) Checkered textile. (f) Knitted fabric. © Cengage Learning 2015.

Figures 15.1a,b and 15.2a,b show that the same number and the same type of primitives do not necessarily give the same texture. Similarly, Figures 15.2a and 15.2c show that the same spatial relationship of primitives does not guarantee texture uniqueness, and therefore is not sufficient for description. Tone and structure are not independent; textures always display both even though one or the other usually dominates, and we usually speak about one or the other only. Tone can be understood as tonal properties of primitives, taking primitive spatial relationships into consideration. Structure refers to spatial relationships of primitives considering their tonal properties as well.

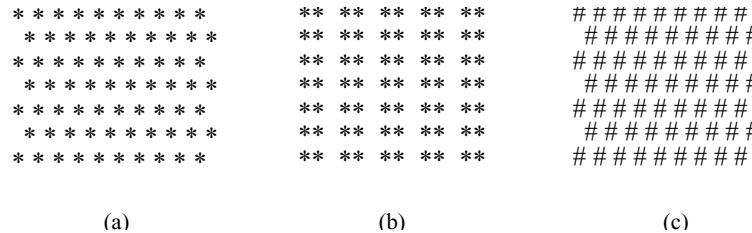


Figure 15.2: Artificial textures. © Cengage Learning 2015.

If the texture primitives are small and if the tonal differences between neighboring primitives are large, a **fine** texture results (Figures 15.1a,b and 15.1d). If the primitives are larger and consist of several pixels, a **coarse** texture results (Figures 15.1c and 15.1e). Again, this is a reason for using both tonal and structural properties in texture description. Note that the fine/coarse texture characteristic depends on scale.

Further, textures can be classified according to their strength—texture strength then influences the choice of description method. **Weak** textures have small spatial interactions between primitives, and can be adequately described by frequencies of primitive types appearing in some neighborhood. Because of this, many statistical properties are evaluated in the description of weak textures. In **strong** textures, the spatial interactions between primitives are somewhat regular. To describe strong textures, the frequency of occurrence of primitive pairs in some spatial relationship may be sufficient. Strong texture recognition is usually accompanied by an exact definition of primitives and their spatial relationships.

It remains to define a constant texture. One existing definition [Sklansky, 1978] suggests ‘an image region has a constant texture if a set of its local properties in that region is constant, slowly changing, or approximately periodic’. The set of local properties can be understood as primitive types and their spatial relationships. An important part of the definition is that the properties must be repeated inside the constant texture area. How many times must the properties be repeated? Assume that a large area of constant texture is available, and consider smaller and smaller parts of that texture, digitizing it at constant resolution as long as the texture character remains unchanged. Alternatively, consider larger and larger parts of the texture, digitizing it at constant raster, until details become blurred and the primitives finally disappear. We see that image resolution (scale) must be a consistent part of the description; if the resolution is appropriate, texture character does not change for any position in our window.

Two main texture description approaches exist—**statistical** and **syntactic** [Haralick, 1979]. Statistical methods are suitable if primitive sizes are comparable with pixel sizes. Syntactic and **hybrid** methods (combinations of statistical and syntactic) are more suit-

able for textures where primitives can be assigned a label—the primitive type—meaning that primitives can be described using a larger variety of properties than just tonal properties; for example, shape description. Instead of tone, brightness will be used more often in the following sections because it corresponds better to gray-level images.

Research on pre-attentive (early) vision [Julesz, 1981] shows that human ability to recognize texture quickly is based mostly on **textons**, which are elongated blobs (rectangles, ellipses, line segments, line ends, crossings, corners) that can be detected by pre-attentive vision, while the positional relationship between neighboring textons must be done slowly by an attentive vision subsystem. As a result of these investigations, methods based on texton detection and texton density computation were developed.

15.1 Statistical texture description

Statistical description methods describe each texture by a feature vector of properties which represents a point in a multi-dimensional feature space. The aim is to find a deterministic or probabilistic decision rule assigning a texture to some specific class (see Chapter 9).

15.1.1 Methods based on spatial frequencies

Measuring spatial frequencies is the basis of a large group of recognition methods. Textural character is in direct relation to the spatial size of the primitives; coarse textures are built from larger primitives, fine textures from smaller primitives. Fine textures are characterized by higher spatial frequencies, coarse textures by lower spatial frequencies.

One of many related spatial frequency methods evaluates the **auto-correlation function of a texture**. In an auto-correlation model, a single pixel is considered a texture primitive, and a primitive tone property is the gray-level. Texture spatial organization is described by the correlation coefficient that evaluates linear spatial relationships between primitives. If the texture primitives are relatively large, the auto-correlation function value decreases slowly with increasing distance, while it decreases rapidly if texture consists of small primitives. If primitives are placed periodically in a texture, the auto-correlation increases and decreases periodically with distance.

Texture can be described using the following algorithm.

Algorithm 15.1: Auto-correlation texture description

1. For an $M \times N$ image, evaluate auto-correlation coefficients for several different values of position difference p, q :

$$C_{ff}(p, q) = \frac{MN}{(M-p)(N-q)} \frac{\sum_{i=1}^{M-p} \sum_{j=1}^{N-q} f(i, j)f(i+p, j+q)}{\sum_{i=1}^M \sum_{j=1}^N f^2(i, j)}. \quad (15.1)$$

2. Alternatively, the auto-correlation function can be determined in the frequency domain from the image power spectrum [Castleman, 1996]:

$$C_{ff} = \mathcal{F}^{-1}\{|F|^2\}. \quad (15.2)$$

If the textures are circularly symmetric, the auto-correlation texture description can be computed as a function of the absolute position difference not considering direction—that is, a function of one variable.

Spatial frequencies can also be determined from an **optical image transform** (recall that the Fourier transform can be realized by a convex lens—see Section 3.2) [Shulman, 1970], a big advantage of which is that it may be computed in real time. The Fourier transform describes an image by its spatial frequencies; average values of energy in specific wedges and rings of the Fourier spectrum can be used as textural description features (see Figure 15.3). Features evaluated from rings reflect coarseness of the texture—high energy in large-radius rings is characteristic of fine textures (high frequencies), while high energy in small radii is characteristic of coarser ones. Features evaluated from wedge slices of the Fourier transform image depend on directional properties—if a texture has many edges or lines in a direction ϕ , high energy will be present in a wedge in direction $\phi + \pi/2$.

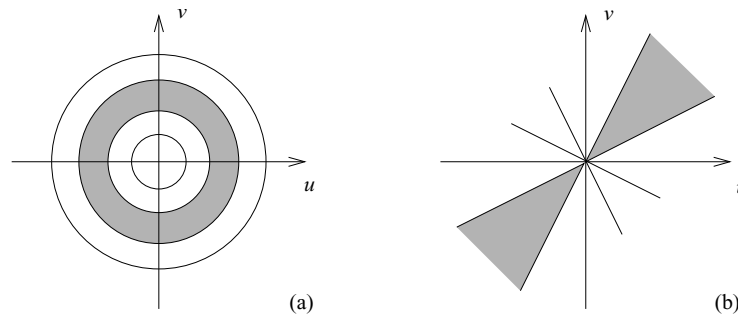


Figure 15.3: Partitioning of Fourier spectrum. (a) Ring filter. (b) Wedge filter reflecting the Fourier spectrum symmetry. © Cengage Learning 2015.

Similarly, a **discrete image transform** may be used. An image is usually divided into small $n \times n$ non-overlapping subimages, and the gray-levels of its pixels may be interpreted as an n^2 -dimensional vector, and an image can be represented as a set of vectors. These vectors are transformed applying a Fourier, Hadamard, or other discrete image transform (Section 3.2). The new co-ordinate system's basis vectors are related to the spatial frequencies of the original image and can be used for texture description [Rosenfeld, 1976]. When description of noisy texture becomes necessary, the problem becomes more difficult. Liu and Jernigan extracted a set of 28 spatial frequency-domain features, and determined a subset insensitive to additive noise: dominant peak energy, power spectrum shape, entropy [Liu and Jernigan, 1990].

Many problems remain with spatial frequency methods—resulting descriptions are not invariant even to monotonic image gray-level transforms, and it can be shown [Weszka et al., 1976] that the frequency-based approach is less efficient than others. A joint spatial/spatial frequency approach is recommended; the Wigner distribution was shown to be useful in a variety of synthetic and Brodatz textures [Reed et al., 1990].

15.1.2 Co-occurrence matrices

Co-occurrence matrix methods are based on the repeated occurrence of some gray-level configuration in the texture; this configuration varies rapidly with distance in fine textures and slowly in coarse textures [Haralick et al., 1973]. Suppose the part of a textured image to be analyzed is an $M \times N$ rectangular window. An occurrence of some gray-level configuration may be described by a matrix of relative frequencies $P_{\phi,d}(a,b)$, describing how frequently two pixels with gray-levels a, b appear in the window separated by a distance d in direction ϕ . These matrices are symmetric if defined as given below. However, an asymmetric definition may be used, where matrix values are also dependent on the direction of co-occurrence. A co-occurrence matrix computation scheme was given in Algorithm 4.1.

Setting $D = (M \times N) \times (M \times N)$, non-normalized frequencies of co-occurrence as functions of angle and distance can be represented formally as

$$\begin{aligned}
 P_{0^\circ,d}(a,b) &= |\{(k,l), (m,n) \in D : \\
 &\quad k-m=0, |l-n|=d, f(k,l)=a, f(m,n)=b\}|, \\
 P_{45^\circ,d}(a,b) &= |\{(k,l), (m,n) \in D : \\
 &\quad (k-m=d, l-n=-d) \vee (k-m=-d, l-n=d), f(k,l)=a, f(m,n)=b\}|, \\
 P_{90^\circ,d}(a,b) &= |\{(k,l), (m,n) \in D : \\
 &\quad |k-m|=d, l-n=0, f(k,l)=a, f(m,n)=b\}|, \\
 P_{135^\circ,d}(a,b) &= |\{(k,l), (m,n) \in D : \\
 &\quad (k-m=d, l-n=d) \vee (k-m=-d, l-n=-d), f(k,l)=a, f(m,n)=b\}|.
 \end{aligned} \tag{15.3}$$

An example will illustrate for the distance $d = 1$. A 4×4 image with four gray-levels is given in Figure 15.4. The element $P_{0^\circ,1}(0,0)$ represents the number of times two pixels

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Figure 15.4: Gray-level image.
© Cengage Learning 2015.

with gray-levels 0 and 0 appear separated by distance 1 in direction 0° ; $P_{0^\circ,1}(0,0) = 4$ in this case. The element $P_{0^\circ,1}(3,2)$ represents the number of times two pixels with gray-levels 3 and 2 appear separated by distance 1 in direction 0° ; $P_{0^\circ,1}(3,2) = 1$. Note that $P_{0^\circ,1}(2,3) = 1$ due to matrix symmetry:

$$P_{0^\circ,1} = \begin{vmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{vmatrix}, \quad P_{135^\circ,1} = \begin{vmatrix} 2 & 1 & 3 & 0 \\ 1 & 2 & 1 & 0 \\ 3 & 1 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{vmatrix}.$$

Texture classification can be based on criteria derived from the following co-occurrence matrices.

- **Energy**, or angular second moment (an image homogeneity measure—the more homogeneous the image, the larger the value)

$$\sum_{a,b} P_{\phi,d}^2(a,b) . \quad (15.4)$$

- **Entropy**:

$$-\sum_{a,b} P_{\phi,d}(a,b) \log_2 P_{\phi,d}(a,b) . \quad (15.5)$$

- **Maximum probability**:

$$\max_{a,b} P_{\phi,d}(a,b) . \quad (15.6)$$

- **Contrast** (a measure of local image variations; typically $\kappa = 2, \lambda = 1$):

$$\sum_{a,b} |a-b|^\kappa P_{\phi,d}^\lambda(a,b) . \quad (15.7)$$

- **Inverse difference moment**:

$$\sum_{a,b;a \neq b} \frac{P_{\phi,d}^\lambda(a,b)}{|a-b|^\kappa} . \quad (15.8)$$

- **Correlation** (a measure of image linearity, linear directional structures in direction ϕ result in large correlation values in this direction):

$$\frac{\sum_{a,b} [(ab)P_{\phi,d}(a,b)] - \mu_x \mu_y}{\sigma_x \sigma_y} , \quad (15.9)$$

where μ_x, μ_y are means and σ_x, σ_y are standard deviations

$$\begin{aligned} \mu_x &= \sum_a a \sum_b P_{\phi,d}(a,b) , & \sigma_x^2 &= \sum_a (a - \mu_x)^2 \sum_b P_{\phi,d}(a,b) , \\ \mu_y &= \sum_b b \sum_a P_{\phi,d}(a,b) , & \sigma_y^2 &= \sum_b (b - \mu_y)^2 \sum_a P_{\phi,d}(a,b) . \end{aligned}$$

Algorithm 15.2: Co-occurrence method of texture description

1. Construct co-occurrence matrices for given directions and given distances.
2. Compute texture feature vectors for four directions ϕ , different values of d , and the six characteristics. This results in many correlated features.

The co-occurrence method describes second-order image statistics and works well for a large variety of textures (see [Gotlieb and Kreyszig, 1990] for a survey of texture descriptors based on co-occurrence matrices). Good properties of the co-occurrence method

are the description of spatial relations between tonal pixels, and invariance to monotonic gray-level transformations. On the other hand, it does not consider primitive shapes, and therefore cannot be recommended if the texture consists of large primitives. Memory requirements were once a disadvantage, although this is much less the case today. The number of gray-levels could be reduced to 32 or 64, decreasing co-occurrence matrix sizes, but loss of gray-level accuracy is a negative effect (although this loss is usually insignificant in practice). Likewise, the method is computationally expensive but this is not a significant constraint on modern hardware.

15.1.3 Edge frequency

Methods discussed so far describe texture by its spatial frequencies, but comparison of textural edge frequencies can be used as well. Edges can be detected either as micro-edges using small edge operator masks, or as macro-edges using large masks [Davis and Mitiche, 1980]—virtually any edge detector can be used (see Section 5.3.2). The distance-dependent texture description function $g(d)$ can be computed for any subimage f defined in a neighborhood N for variable distance d :

$$g(d) = |f(i, j) - f(i + d, j)| + |f(i, j) - f(i - d, j)| + |f(i, j) - f(i, j + d)| + |f(i, j) - f(i, j - d)|. \quad (15.10)$$

The function $g(d)$ is similar to the negative auto-correlation function; its minimum (resp. maximum) corresponds to the maximum (minimum) of the auto-correlation function.

Algorithm 15.3: Edge-frequency texture description

1. Compute a gradient $g(d)$ for all pixels of the texture.
2. Evaluate texture features as average values of gradient in specified distances d .

Dimensionality of the texture description feature space is then given by the number of distance values d used to compute the edge gradient.

Several other texture properties may be derived from first-order and second-order statistics of edge distributions [Tomita and Tsuji, 1990].

- **Coarseness:** Edge density is a measure of coarseness. The finer the texture, the higher the number of edges present in the texture edge image.
- **Contrast:** High-contrast textures are characterized by large edge magnitudes.
- **Randomness:** This is measured as entropy of the edge magnitude histogram.
- **Directivity:** An approximate measure of directivity may be determined as entropy of the edge direction histogram. Directional textures have an even number of significant histogram peaks, directionless textures have a uniform edge direction histogram.
- **Linearity:** Texture linearity is indicated by co-occurrences of edge pairs with the same edge direction at constant distances, and edges are positioned in the edge direction (see Figure 15.5, edges a and b).

- **Periodicity:** Texture periodicity can be measured by co-occurrences of edge pairs of the same direction at constant distances in directions perpendicular to the edge direction (see Figure 15.5, edges *a* and *c*).
- **Size:** Texture size may be based on co-occurrences of edge pairs with opposite edge directions at constant distance in a direction perpendicular to the edge directions (see Figure 15.5, edges *a* and *d*).

Note that the first three measures are derived from first-order statistics, the last three from second-order statistics.

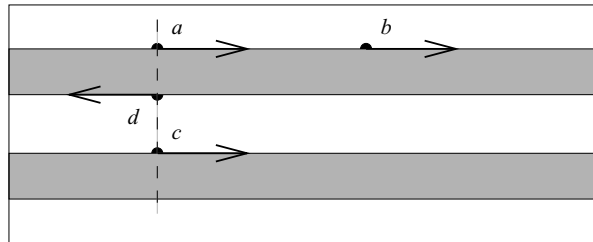


Figure 15.5: Texture linearity, periodicity, and size measures may be based on image edges. Based on [Tomita and Tsuji, 1990].

Many existing texture recognition methods are based on texture detection. The concepts of pre-attentive vision and textons have been mentioned, which are also based mostly on edge-related information. A zero-crossing operator was applied to edge-based texture description in [Perry and Lowe, 1989]; the method determines image regions of a constant texture, assuming no a priori knowledge about the image, texture types, or scale. Feature analysis is performed across multiple window sizes.

A slightly different approach to texture recognition may require detection of borders between homogeneous textured regions. A hierarchical algorithm for textured image segmentation is described in [Fan, 1989], and a two-stage contextual classification and segmentation of textures based on a coarse-to-fine principle of edge detection is given in [Fung et al., 1990]. Texture description and recognition in the presence of noise represents a difficult problem. A noise-tolerant texture classification approach based on a Canny-type edge detector is discussed in [Kjell and Wang, 1991] where texture is described using periodicity measures derived from noise-insensitive edge detection.

15.1.4 Primitive length (run length)

A large number of neighboring pixels of the same gray-level represents a coarse texture, pixels represents a fine texture, and the lengths of texture primitives in different directions can serve as a texture description [Galloway, 1975]. A **primitive** is a maximum contiguous set of constant-level pixels located in a line; these can then be described by gray-level, length, and direction. Texture description features can be based on computation of continuous probabilities of the length and gray-level of primitives in the texture.

Let $B(a, r)$ be the number of primitives of all directions having length r and gray-level a , M, N the image dimensions, and L the number of image gray-levels. Let N_r be

the maximum primitive length in the image, and let K be the total number of runs

$$K = \sum_{a=1}^L \sum_{r=1}^{N_r} B(a, r) . \quad (15.11)$$

Then:

- **Short primitives emphasis:**

$$\frac{1}{K} \sum_{a=1}^L \sum_{r=1}^{N_r} \frac{B(a, r)}{r^2} . \quad (15.12)$$

- **Long primitives emphasis:**

$$\frac{1}{K} \sum_{a=1}^L \sum_{r=1}^{N_r} B(a, r) r^2 . \quad (15.13)$$

- **Gray-level uniformity:**

$$\frac{1}{K} \sum_{a=1}^L \left[\sum_{r=1}^{N_r} B(a, r) \right]^2 . \quad (15.14)$$

- **Primitive length uniformity:**

$$\frac{1}{K} \sum_{r=1}^{N_r} \left[\sum_{a=1}^L B(a, r) \right]^2 . \quad (15.15)$$

- **Primitive percentage:**

$$\frac{K}{\sum_{a=1}^L \sum_{r=1}^{N_r} r B(a, r)} = \frac{K}{MN} . \quad (15.16)$$

A general algorithm might then be the following.

Algorithm 15.4: Primitive-length texture description

1. Find primitives of all gray-levels, all lengths, and all directions.
2. Compute texture description features as given in equations (15.12–15.16). These features then provide a description vector.

15.1.5 Laws' texture energy measures

Laws' texture energy measures determine texture properties by assessing average gray-level, edges, spots, ripples, and waves [Laws, 1979; Wu et al., 1992]. They are derived from three simple vectors: $L_3 = (1, 2, 1)$ (averaging); $E_3 = (-1, 0, 1)$ (first difference, edges); and $S_3 = (-1, 2, -1)$ (second difference, spots). After convolution of these vectors with themselves and each other, five vectors result:

$$\begin{aligned} L_5 &= (1, 4, 6, 4, 1), \\ E_5 &= (-1, -2, 0, 2, 1), \\ S_5 &= (-1, 0, 2, 0, -1), \\ R_5 &= (1, -4, 6, -4, 1), \\ W_5 &= (-1, 2, 0, -2, -1). \end{aligned} \quad (15.17)$$

Mutual multiplying of these vectors, considering the first term as a column vector and the second term as a row vector, results in 5×5 Laws' masks. For example,

$$L_5^T \times S_5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}. \quad (15.18)$$

By convolving the masks with an image and calculating energy statistics, a feature vector is derived that can be used for texture description.

15.1.6 Local binary patterns—LBPs

Local binary pattern (LBP) approach were introduced by Ojala et al. [Ojala et al., 1996], motivated by three-valued texture units originally presented by Wang and He [Wang and He, 1990]. The main idea of the LBP approach—as its name reflects—is to locally threshold the brightness of a pixel's neighborhood at the center pixel gray level to form a binary pattern. The LBP description given here is based on [Ojala et al., 1996; Pietikainen et al., 2000; Ojala et al., 2001, 2002b].

The LBP operator is gray-scale invariant and is derived as follows: texture is described in a local neighborhood of a central pixel, the neighborhood consisting of P ($P > 1$) equally spaced points on a circle of radius $R > 0$ centered at the center pixel. The texture is described as a joint distribution

$$T = t(g_c, g_0, g_1, \dots, g_{P-1}), \quad (15.19)$$

where g_c is the gray level of the central pixel and $g_0, \dots, g_{P-1}$ are gray values of the neighborhood pixels. Assuming the coordinates of G_c are $(0,0)$, coordinates of the neighborhood pixels g_p are given by $[-R \sin(2\pi p/P), R \cos(2\pi p/P)]$. If a point does not fall exactly at the center of a pixel, its value is estimated by interpolation (Figure 15.6).

Using this texture representation, gray-scale invariance can be achieved using gray-level differences rather than brightness values:

$$T = t(g_c, g_0 - g_c, g_1 - g_c, \dots, g_{P-1} - g_c). \quad (15.20)$$

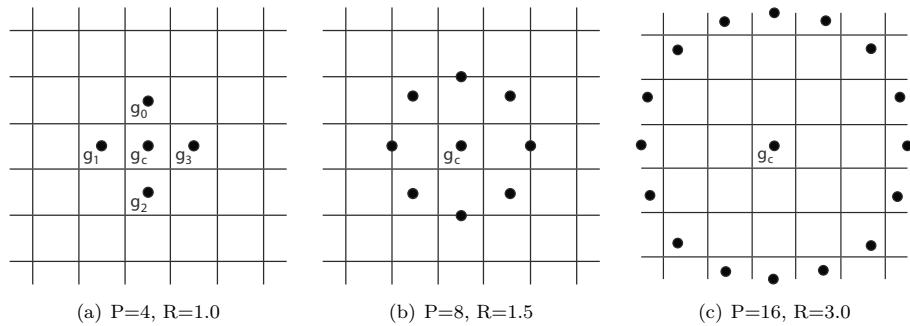


Figure 15.6: Circularly symmetric neighborhoods for different values of P and R [Ojala et al., 2002b]. © Cengage Learning 2015.

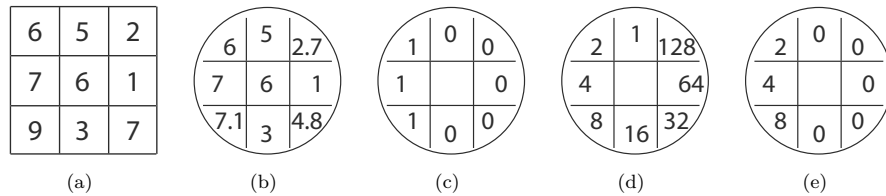


Figure 15.7: Binary texture description operator $LBP_{8,1}$. (a) Original gray values of a 3×3 image. (b) Gray-level interpolation achieves symmetric circular behavior. Linear interpolation was used for simplicity (Section 5.2.2). (c) Circular operator values after binarization, equations (15.23–15.24). (d) Directional weights. (e) Directional values associated with $LBP_{8,1}$ —the resulting value of $LBP_{8,1} = 14$. If rotationally normalized, the weighting mask would rotate by one position counterclockwise, yielding $LBP_{8,1}^i = 7$. © Cengage Learning 2015.

Assuming that brightness g_c is independent of the differences $g_p - g_c$ (not quite valid but close-enough to permit approximation), texture can be represented as:

$$T \approx l(g_c)t(g_0 - g_c, g_1 - g_c, \dots, g_{P-1} - g_c), \quad (15.21)$$

where the image luminance is represented by $l(g_c)$ and the texture by the brightness differences between the central and neighboring pixels. Since image luminance itself does not contribute to texture properties, texture description can be based on differences only:

$$T \approx t(g_0 - g_c, g_1 - g_c, \dots, g_{P-1} - g_c). \quad (15.22)$$

Texture description is given by calculating occurrences of neighborhood brightness patterns in a P -layer histogram, each layer corresponding to one of P direction for given value of R . Clearly, all differences are zero for a constant-brightness region, high in all directions for a spot located at g_c , and exhibit varying values along local image edges. This histogram can be used for texture discrimination.

This description is invariant to brightness shifts. To achieve invariance to brightness scaling, the absolute values of gray level differences may be replaced with their signs as shown in Figure 15.7a,b.

$$T \approx t(s(g_0 - g_c), s(g_1 - g_c), \dots, s(g_{P-1} - g_c)), \quad (15.23)$$

where

$$s(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}. \quad (15.24)$$

When the operator elements are ordered to form a circular chain with values of zero and one, then specific directions can be consistently weighted forming a scalar chain code descriptor. The chain code contributors can be summed over the entire circular neighborhood of P pixels as depicted on Figure 15.7c,d and the local texture pattern can be described by a single number for any specific (P, R) combination. Weights 2^p can be assigned in a circular fashion with p increasing for all P points:

$$\text{LBP}_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p. \quad (15.25)$$

For a texture patch, these $\text{LBP}_{P,R}$ values can be used to form single- or multi-dimensional histograms or feature vectors, or can be further processed to become rotation and/or spatial scale invariant as described below.

When the image is rotated, image gray values travel around the circle, affecting the LBP values. To achieve rotational invariance it is natural to normalize the circular chain code in a way that minimizes the resulting LBP^{ri} value (Figure 15.7, see also Section 8.2.1):

$$\text{LBP}_{P,R}^{ri} = \min_{i=0,1,\dots,P-1} \{ \text{ROR}(\text{LBP}_{P,R}, i) \}, \quad (15.26)$$

where $\text{ROR}(x, i)$ denotes a circular bitwise right shift on the P -bit number x i -times—or simply rotating the circular neighbor set clockwise so that the resulting LBP value is minimized (bitwise operation instructions for right (ROR) and left (ROL) circular shifts exist in Intel x86 instruction set). Patterns $\text{LBP}_{P,R}^{ri}$ can be used as feature detectors—for $\text{LBP}_{8,R}^{ri}$, 36 such feature detectors can be formed as shown in Figure 15.8. Pattern #0 would indicate a bright spot location, #8 a dark spot location or a flat area, #4 corresponds to straight edges, etc.

While the $\text{LBP}_{8,R}^{ri}$ features do not perform very well in real-world problems [Pietikainen et al., 2000], local binary patterns can be derived from them which start representing fundamental texture properties. Such derived LBP's are called **uniform patterns** due to having uniform circular structure with minimal spatial transitions. For $\text{LBP}_{8,R}^{ri}$, such uniform patterns are shown in the first row of Figure 15.8. The uniform patterns can be considered **microstructure templates** with the same interpretation as given above—#0 being a bright spot microtemplate, etc. Formally, a **uniformity measure** U can be introduced, reflecting the number of 0/1 (or 1/0) transitions. Note that all the uniform patterns have U values of 2 or less, while all other patterns have a U value of at least 4. Subsequently, a gray-scale and rotation invariant texture descriptor is defined as

$$\text{LBP}_{P,R}^{riu2} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & \text{if } U(\text{LBP}_{P,R}) \leq 2 \\ P + 1 & \text{otherwise} \end{cases}, \quad (15.27)$$

where

$$U(\text{LBP}_{P,R}) = |s(g_{P-1} - g_c) - s(g_0 - g_c)| + \sum_{p=1}^{P-1} |s(g_p - g_c) - s(g_{p-1} - g_c)|. \quad (15.28)$$

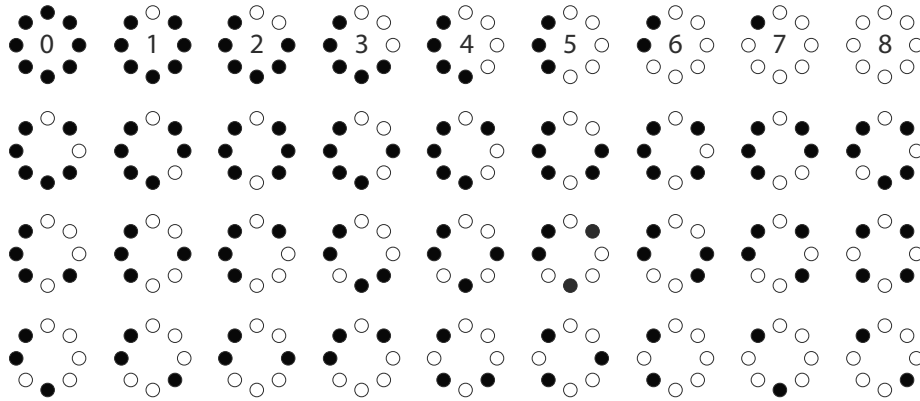


Figure 15.8: For example, when considering $P = 8$ and any value of R , 36 unique circularly symmetric feature detectors can be formed: black and white circles correspond to bit values of 0 and 1. The first row shows the 9 “uniform” patterns with their $\text{LBP}_{8,R}^{\text{riu2}}$ values given (equation 15.27). Based on [Ojala et al., 2002b].

Here, the superscript *riu2* denotes rotational invariant uniform patterns with uniformity values of at most 2. Note that only $P+2$ patterns can exist: $P+1$ uniform patterns and one additional ‘catch-all’ pattern (Figure 15.8). Mapping from $\text{LBP}_{P,R}$ to $\text{LBP}_{P,R}^{\text{riu2}}$ is best implemented using a look-up table with 2^P elements.

Texture description is based on a histogram of $\text{LBP}_{P,R}^{\text{riu2}}$ operator outputs accumulated over a texture patch. The reason this approach works much better than using $\text{LBP}_{P,R}^i$ features directly is associated with the overwhelmingly larger proportion of uniform patterns when collecting the microstructure templates. Due to their relatively low occurrence frequencies, statistical properties of ‘non-uniform’ patterns cannot be reliably estimated and resulting noisy estimates negatively influence texture discrimination. For example, when analyzing Brodatz textures, $\text{LBP}_{8,1}^i$ features consist of 87% uniform and only 13% non-uniform patterns. Since only 9 uniform templates exist while three times as many (27) non-uniform templates can be formed, the frequency differences become even more striking. Similarly, the uniform/non-uniform frequency distributions are 67–33% for $\text{LBP}_{16,2}^i$ and 50–50% for $\text{LBP}_{24,3}^i$ on the same set of textures. These distributions seem quite stable across different texture discrimination problems [Ojala et al., 2002b].

The choice of P and R remains to be considered. Increasing P helps with overcoming the crudeness of angular quantization. Clearly, P and R are related in the sense that the radius must increase proportionally with denser angular sampling, or the number of non-redundant pixel values in the circular neighborhood will become a limiting factor (nine non-redundant pixels are available for $R = 1$). At the same time, if P is increased too much, the size 2^P of the look-up table will affect computational efficiency. Practical experiments limited P values to 24 [Ojala et al., 2002b], resulting in a 16MB look-up table, an easily manageable size. Additional aspects of rotation-invariant variance measure and multi-resolution analysis of textures are all discussed in [Ojala et al., 2002b].

When using LBP features and pattern histograms for texture classification, non-parametric statistical tests were employed to determine dissimilarity of the histogram description from all class-specific histograms of LBP features obtained during training.

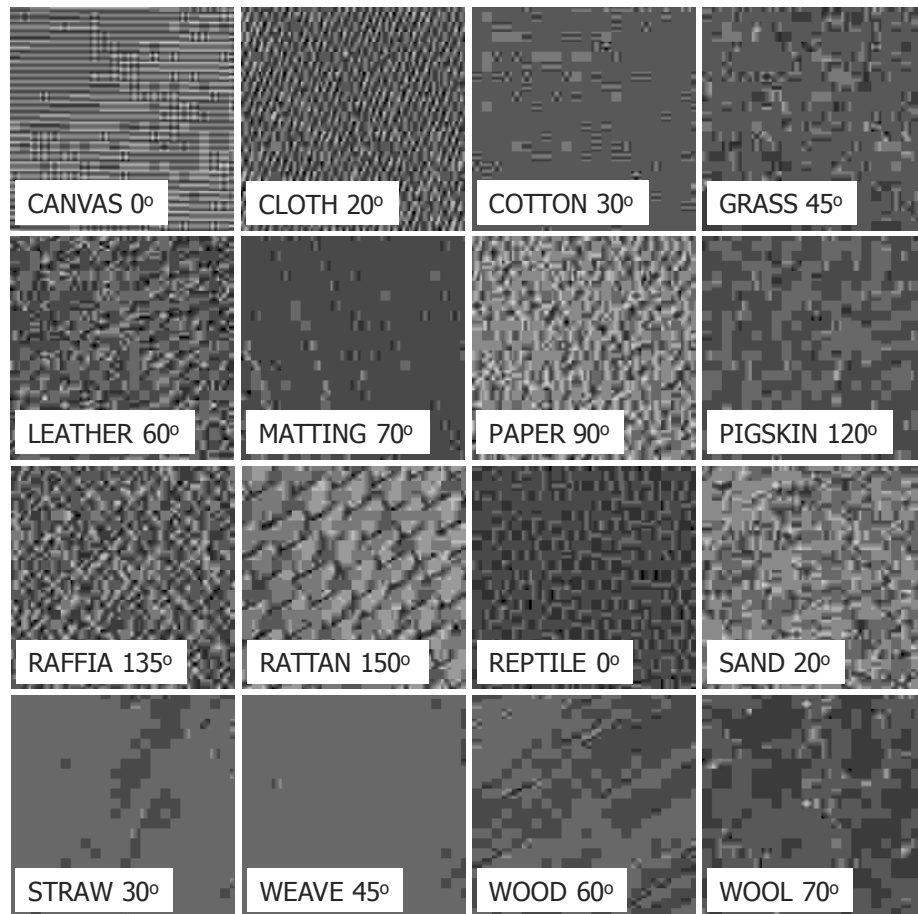


Figure 15.9: Samples of 16 Brodatz textures used for $LBP_{P,R}^{riu2}$ evaluation in [Ojala et al., 2002b]. Patches shown are 180×180 pixels and were rotated at different angles in addition to the angular rotations depicted in the figure. *Courtesy of M. Pietikainen, T. Ojala, Oulu University, Finland.*

The lowest (and perhaps below-minimum threshold) dissimilarity criterion identifies the most likely texture class the patch sample belongs to. This has an additional advantage of permitting an ordering of the most likely classifications according to their likelihood. Non-parametric statistical tests like chi-square or G (log-likelihood ratio) can be used to assess the goodness of fit.

When this approach was applied to classification of 16 Brodatz textures (Figure 15.9), the $LBP_{P,R}^{riu2}$ histograms, followed by goodness-of-fit analysis, outperformed wavelet transforms, Gabor transforms, and Gaussian Markov Random Field approaches while exhibiting the lowest computational complexity. Rotational invariance was demonstrated by training the LBP method in textures of single orientation and testing independent samples rotated using 6 different angles (Figure 15.9). $LBP_{8,1}^{riu2}$, $LBP_{16,2}^{riu2}$, and $LBP_{24,3}^{riu2}$ were used and 100% classification was achieved for some feature combinations including

variance measures, compared with the second best 95.8% reported in [Porter and Canagarajah, 1997], achieved using wavelets. Another set of experiments used 24 classes of natural textures acquired using a robotic arm-mounted camera at different angles and with varying controlled illumination. The LBP method demonstrated excellent performance. Test image data and the texture classification software test suite *OUTEX* can be accessed at <http://www.outex.oulu.fi/> [Ojala et al., 2002a].

An interesting adaption of these ideas has constructed LBPs of gradient images to assist in face recognition [Vu et al., 2012]. The approach—in which gradient LBPs were supplemented by Gaussian mixture model (GMMs, Section 10.13) and support vector machines (SVMs, Section 9.2.5)—proves very fast and efficient, outperforming comparable techniques in performance as well.

An extension of LBPs called **local ternary patterns** or LTPs, takes advantage of creating a third pattern in equation (15.24) [Tan and Triggs, 2010]. LTPs allow three labels to form an insensitivity interval around the value of the central pixel:

$$s'(x) = \begin{cases} 1 & \text{for } x > t \\ 0 & \text{for } x \geq -t \text{ and } x \leq t \\ -1 & \text{for } x < -t \end{cases}, \quad (15.29)$$

where t is an insensitivity threshold. Compared to LBPs, LTPs were found more discriminant and less sensitive to noise in uniform regions. Note however, that gray-level transform invariance is somewhat compromised.

15.1.7 Fractal texture description

Fractal-based texture analysis was introduced in [Pentland, 1984], where a correlation between texture coarseness and fractal dimension of a texture was demonstrated. A fractal is defined [Mandelbrot, 1982] as a set for which the Hausdorff-Besicovich dimension [Hausdorff, 1919; Besicovitch and Ursell, 1937] is strictly greater than the topological dimension; therefore, fractional dimension is the defining property. Fractal models typically relate a metric property such as line length or surface area to the elementary length or area used as a basis for determining the metric property—measuring coast length is a frequently used example [Pentland, 1984; Lundahl et al., 1986]. Suppose coastal length is determined by applying a 1-km-long ruler end-to-end to the coastline; the same procedure can be repeated with a 0.5-km ruler, and other shorter or longer rulers. It is easy to see that shortening the ruler will be associated with an increase in total length. Importantly, the relation between the ruler length and the measured coast length can be considered a measure of the coastline's geometric properties, e.g., its roughness. The functional relationship between the ruler size r and the measured length L can be expressed as

$$L = cr^{1-D}, \quad (15.30)$$

where c is a scaling constant and D is the **fractal dimension** [Mandelbrot, 1982]. Fractal dimension has been shown to correlate well with the function's intuitive roughness.

While equation (15.30) can be applied directly to lines and surfaces, it is often more appropriate to consider the function as a stochastic process. One of the most important stochastic fractal models is the *fractional Brownian motion model* [Mandelbrot, 1982], which considers naturally rough surfaces as the end results of random walks. Importantly,

intensity surfaces of textures can also be considered as resulting from random walks, and the fractional Brownian motion model can be applied to texture description.

Fractal description of textures is typically based on determination of fractal dimension and **lacunarity** to measure texture roughness and granularity from the image intensity function. The topological dimension of an image is equal to three—two spatial dimensions and the third representing the image intensity. Considering the topological dimension T_d , the fractal dimension D can be estimated from the Hurst coefficient H [Hurst, 1951; Mandelbrot, 1982] as

$$D = T_d - H. \quad (15.31)$$

For images ($T_d = 3$), the Hurst parameter H or the fractal dimension D can be estimated from the relationship

$$E((\Delta f)^2) = c[(\Delta r)^H]^2 = c(\Delta r)^{6-2D}, \quad (15.32)$$

where $E()$ is an expectation operator, $\Delta f = f(i, j) - f(k, l)$ is the intensity variation, c is a scaling constant, and $\Delta r = \|(i, j) - (k, l)\|$ is the spatial distance. A simpler way to estimate fractal dimension is to use the equation:

$$E(|\Delta f|) = \kappa(\Delta r)^{3-D}, \quad (15.33)$$

where $\kappa = E(|\Delta f|)_{\Delta r=1}$. By taking logarithms and noting that $H = 3 - D$,

$$\log E(|\Delta f|) = \log \kappa + H \log(\Delta r). \quad (15.34)$$

The parameter H can be obtained by using least-squares linear regression to estimate the slope of the curve of gray-level differences $gd(k)$ versus κ in log-log scales [Wu et al., 1992]. Considering an $M \times M$ image f ,

$$gd(k) = \frac{1}{\mu} \left(\sum_{i=0}^{M-1} \sum_{j=0}^{M-k-1} |f(i, j) - f(i, j+k)| + \sum_{i=0}^{M-k-1} \sum_{j=0}^{M-1} |f(i, j) - f(i+k, j)| \right), \quad (15.35)$$

where $\mu = 2M(M - k - 1)$. The scale k varies from 1 to the maximum selected value s . Fractal dimension D is then derived from the value of the Hurst coefficient. The approximation error of the regression line fit should be determined to demonstrate that the texture is a fractal, and can thus be efficiently described using fractal measures. A small value of the fractal dimension D (large value of the parameter H) represents a fine texture, while large D (small H) corresponds to a coarse texture.

Single fractal dimension is not sufficient for description of natural textures. Lacunarity measures describe characteristics of textures of different visual appearance that have the same fractal dimension [Voss, 1986; Keller et al., 1989; Wu et al., 1992]. Given a fractal set A , let $P(m)$ represent the probability that there are m points within a box of size L centered about an arbitrary point of A . Let N be the number of possible points within the box, then $\sum_{m=1}^N P(m) = 1$ and the lacunarity λ is defined as

$$\lambda = \frac{M_2 - M^2}{M^2}, \quad (15.36)$$

where

$$M = \sum_{m=1}^N m P(m), \quad M_2 = \sum_{m=1}^N m^2 P(m). \quad (15.37)$$

Lacunarity represents a second-order statistic and is small for fine textures and large for coarse ones.

A multi-resolution approach to fractal feature extraction was introduced in [Wu et al., 1992]. The multi-resolution feature vector MF that describes both texture roughness and lacunarity is defined as

$$MF = (H^{(m)}, H^{(m-1)}, \dots, H^{(m-n+1)}), \quad (15.38)$$

where the parameters $H^{(k)}$ are estimated from pyramidal images $f^{(k)}$, where $f^{(m)}$ represents the full-resolution image of size $M = 2^m$, $f^{(m-1)}$ is the half-resolution image of size $M = 2^{m-1}$, etc., and n is the number of resolution levels considered. The multi-resolution feature vector MF can serve as a texture descriptor. Textures with identical fractal dimensions and different lacunarities can be distinguished, as was shown by classification of ultrasonic liver images in three classes—normal, hepatoma, and cirrhosis [Wu et al., 1992]. Practical considerations regarding calculation of fractal-based texture description features can be found in [Sarkar and Chaudhuri, 1994; Huang et al., 1994; Jin et al., 1995].

15.1.8 Multiscale texture description—wavelet domain approaches

Texture description is highly scale dependent. To decrease scale sensitivity, a texture may be described in multiple resolutions and an appropriate scale chosen to achieve maximal discrimination [Unser and Eden, 1989]. **Gabor transforms** and **wavelets** (Section 3.2.7) are well suited to multi-scale characterization [Coggin and Jain, 1985; Mallat, 1989; Bovik et al., 1990; Unser, 1995]. Both are multi-scale spatial—spatial frequency filtering approaches, which were in the past dominated by Gabor filters. Wavelets have been successfully applied to texture classification using pyramid- or tree-structured discrete wavelet transforms [Mallat, 1989; Chang and Kuo, 1993] (Section 3.2.7), typically outperforming conventional texture characterization approaches.

Unser showed that overcomplete **discrete wavelet frames** outperform standard critically sampled wavelet texture feature extraction [Unser, 1995]—what follows is based on this work. Considering a discrete version of the wavelet transform in l_2 (the space of square summable sequences [Rioul, 1993]) textures are described using orthogonal wavelet frames. First, consider the principle of this approach using a single-dimensional signal x . Take a prototype filter h satisfying

$$H(z)H(z^{-1}) + H(-z)H(-z^{-1}) = 1, \quad (15.39)$$

where $H(z)$ is the z -transform of h [Oppenheim et al., 1999], and let the filter also be subjected to the lowpass constraint $H(z)|_{z=1} = 1$. A complementary high-pass filter g is then obtained by shift and modulation

$$G(z) = zH(-z^{-1}). \quad (15.40)$$

Using these two prototypes, a sequence of filters of increasing width can be iteratively generated as follows:

$$H_{i+1}(z) = H(z^{2^i}) H_i(z), \quad (15.41)$$

$$G_{i+1}(z) = G(z^{2^i}) H_i(z), \quad (15.42)$$

for $i = 0, \dots, I-1$, initialized with $H_0(z) = 1$. The filters represent perfect reconstruction filter banks, which are used for definition of the individual wavelets used below. In the signal domain, a two-scale relationship is obtained:

$$\begin{aligned} h_{i+1}(k) &= [h]_{\uparrow 2^i} * h_i(k), \\ g_{i+1}(k) &= [g]_{\uparrow 2^i} * h_i(k), \end{aligned} \quad (15.43)$$

where $[.]_{\uparrow m}$ represents upsampling by a factor of m . In general, each iteration dilates the filters h_i and g_i by a factor of two; this sequence of filters is used to decompose the signal in subbands of approximately one octave each. Importantly, such filter sequences provide a full coverage of the frequency domain.

An orthogonal wavelet decomposition obtained by such a sequence of filters yields discrete normalized basis functions

$$\varphi_{i,l}(k) = 2^{i/2} h_i(k - 2^i l), \quad (15.44)$$

$$\psi_{i,l}(k) = 2^{i/2} g_i(k - 2^i l), \quad (15.45)$$

where i and l are scale and translation indices, respectively; the product $2^{i/2}$ is for inner product normalization. Consider a sequence of nested subspaces $l_2 \supset V_0 \supset V_1 \supset \dots \supset V_I$. Here, $V_i = \text{span}\{\varphi_{i,l}\}_{l \in \mathbb{Z}}$ is the approximation space at resolution i . Let subspaces W_i ($i = 1, \dots, I$) represent residue space at resolution i , defined as an orthogonal complement of V_i with respect to V_{i-1} , i.e., $V_{i-1} = V_i + W_i$ and $V_i \perp W_i$. The minimum l_2 -norm approximation of x at scale i which corresponds to the orthogonal projection into V_i is given by

$$x_{(i)}(k) = \sum_{l \in \mathbb{Z}} s_{(i)}(l) \varphi_{i,l}, \quad (15.46)$$

$$s_{(i)}(l) = \langle x(k), \varphi_{i,l}(k) \rangle_{l_2}, \quad (15.47)$$

where $\langle \cdot, \cdot \rangle$ represents the standard l_2 inner product and $\varphi_{i,0}(k) = 2^{i/2} h_i(k)$ is the discrete scaling function at resolution i . The residue (projection of x into W_i), is given by the complementary wavelet expansion

$$x_{(i-1)}(k) - x_{(i)}(k) = \sum_{l \in \mathbb{Z}} d_{(i)}(l) \psi_{i,l}, \quad (15.48)$$

$$d_{(i)}(l) = \langle x(k), \psi_{i,l}(k) \rangle_{l_2}, \quad (15.49)$$

where $\psi_{i,0}(k) = 2^{i/2} g_i(k)$ is the discrete wavelet at scale i .

By combining the residues over all scales to a given depth I , a full discrete wavelet expansion of the signal is obtained

$$x(k) = \sum_{l \in \mathbb{Z}} s_{(I)}(l) \varphi_{I,l} + \sum_{i=1}^I \sum_{l \in \mathbb{Z}} d_{(i)}(l) \psi_{i,l}, \quad (15.50)$$

where d_i are wavelet coefficients and s_I are expansion coefficients of a coarser approximation $x_{(I)}$, see equation (15.46).

Importantly for texture analysis, equations (15.47) and (15.49) can be obtained by simple filtering and down-sampling, yielding

$$\begin{aligned} s_{(I)}(l) &= 2^{I/2} [h_I^T * x]_{\downarrow 2^I}(l), \\ d_{(i)}(l) &= 2^{i/2} [g_i^T * x]_{\downarrow 2^i}(l), \end{aligned} \quad (15.51)$$

where $i = 1, \dots, I$; $h^T(k) = h(-k)$; and $[\cdot]_{\downarrow m}$ is down-sampling by a factor of m . An efficient algorithm based on direct filtering with a **filter bank** exists and is typically used for this purpose [Mallat, 1989].

The most frequently used discrete wavelet transform features for texture analysis are **wavelet energy signatures** [Chang and Kuo, 1993; Mojsilovic et al., 2000; Arivazhagan and Ganesan, 2003] and their second-order statistics [Van de Wouwer et al., 1999]. While the wavelet-based features can be used directly for texture description, this suffers from the lack of translation invariance which—as described earlier—is an important property in describing texture. One possibility to overcome this limitation is to compute the discrete wavelet transform for all possible shifts of the input signal, giving the following decomposition formulae:

$$\begin{aligned} s_I^{\text{DWF}}(k) &= \langle h_I(k-l), x(k) \rangle_{l_2} = h_I^T * x(k), \\ d_i^{\text{DWF}}(k) &= \langle g_i(k-l), x(k) \rangle_{l_2} = g_i^T * x(k), \end{aligned} \quad (15.52)$$

where $i = 1, \dots, I$ and DWF denotes a discrete wavelet frame representation to distinguish this equation from the earlier one (equation 15.51). Hereon, we will deal with this DWF representation without specifically using the superscript. Equation (15.52) represents a non-sampled version of equation (15.51). The wavelet frame coefficients can be used for translation-invariant texture description. Importantly, a simple reconstruction formula exists and both decomposition and reconstruction can be obtained using filter banks [Unser, 1995].

Practical implementation is based on the two-scale relationship given in equation (15.43) yielding a fast iterative decomposition algorithm

$$\begin{aligned} s_{i+1}(k) &= [h]_{\uparrow 2^i} * s_i(k), \\ d_{i+1}(k) &= [g]_{\uparrow 2^i} * s_i(k), \end{aligned} \quad (15.53)$$

where $i = 0, \dots, I$, with the initial condition $s_0 = x$ (Figure 15.10). A convolution with the basic filters h and g is repeated in each step, yielding an approach the complexity of which is identical in each step and proportional to the number of samples.

Extending the single-dimensional case described above to higher-dimensional signals (to allow its use for texture image analysis) calls for the use of a tensor product formulation [Mallat, 1989]. In a two-dimensional image, four distinct basis functions (four distinct filters) are defined corresponding to the different cross-products of the one-dimensional functions φ and ψ . The decomposition can therefore be obtained by successive one-dimensional processing along the rows and columns of an image. The output of the filter bank given in equation (15.52) can be rearranged in an N -component vector where N is the number of subbands

$$\mathbf{y}(k, l) = (y_i(k, l))_{i=1, \dots, N} = [s_I(k, l); d_I(k, l); \dots; d_1(k, l)]^T. \quad (15.54)$$

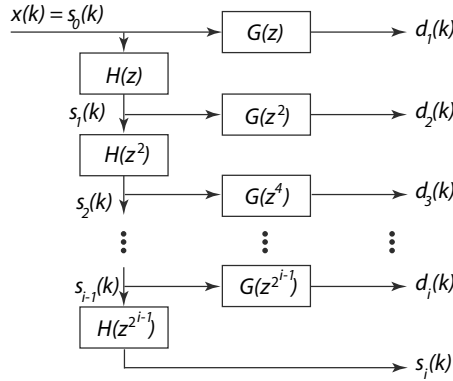


Figure 15.10: Fast iterative approach to discrete wavelet decomposition. © Cengage Learning 2015.

For a spatial coordinate pair (k, l) , the resulting $\mathbf{y}(k, l)$ is a linear transformation of the input vector $\mathbf{x}(k, l)$, which is a block representation of the input image centered at (k, l) . Applying the 2D separable wavelet transform with a depth I yields $N = 1 + 3I$ features.

The texture can consequently be described by the set of N first-order probability density functions $p(y_i)$, for $i = 1, \dots, N$. Also, a more compact representation can be obtained by using a set of channel variance features

$$v_i = \text{var}\{y_i\} \quad (15.55)$$

(see [Unser, 1986] for justification of this approach). Needless to say, texture description capabilities of this methods depend on the appropriate choice of filter bank.

The channel variances v_i can be estimated from average sum of squares over a region of interest R of the analyzed texture

$$v_i = \frac{1}{N_R} \sum_{(k,l) \in R} y_i^2(k, l), \quad (15.56)$$

where N_R is the number of pixels in region R . As mentioned above, the lowpass condition requires $H(z)|_{z=1} = 1$, which in turn yields $E\{y_1\} = E\{x\}$, and $E\{y_i\} = 0$ for $i = 1, \dots, N$. It is therefore recommended to subtract $E\{x\}^2$ from the lowpass channel feature to obtain a better estimate of the variance.

If the discrete wavelet transform is employed, a smaller number of coefficients results due to subsampling. Nevertheless, the variance estimates can be obtained in the same manner. However, an adverse effect on texture classification performance can be observed due to a resulting increased variability of features.

An assessment of the performance of the wavelet domain multiscale approach described above is given in [Unser, 1995]. In experiments performed on 256×256 Brodatz textures [Brodatz, 1966], wavelet and filter bank decompositions were performed by global processing of the images. The performance was assessed in 64 (8×8) non-overlapping subregions sized 32×32 pixels in which the texture was described using independent feature vectors $\mathbf{v} = (v_1, \dots, v_N)$ evaluated according to equation (15.56). Performance assessment demonstrated that the *discrete* wavelet frame approach always outperformed the *discrete* wavelet transform approach. It shows that true multi-resolution feature extraction with two or three levels ($I = 2, 3$) is preferable to local single-scale analysis.

The results also showed that even with $n = 0$, the DWF features performed very well. Importantly, the multiscale approach with 3 decompositions ($n = 3$) at 3 levels of scale ($I = 3$) and using 10 features outperformed (correctness of 99.2%) the single scale DWF approach ($n = 0$, $I = 1$) which used 4 features (correctness of 96.5%). This is notable since other comparison studies previously demonstrated that this DWF approach ($n = 0$, $I = 1$, equivalent to local linear transform using a 2×2 Hadamard transform [Unser, 1986]) typically outperforms most other statistical texture description methods including co-occurrence matrices, correlation, etc. and can thus be used as a reference method for single-scale analysis. The studies in [Unser, 1995] also compare performance of various orthogonal and non-orthogonal wavelet transforms.

Comparison of texture classification behavior of Gabor transforms and wavelets is given in [Vautrot et al., 1996]. If texture segmentation is required, a coarse-to-fine multi-resolution strategy may be used. Approximate border positions between texture regions are detected first in a low-resolution image, accuracy being improved in higher resolutions using the low-level segmentation as a priori information. Wavelet-domain hidden Markov models (Section 10.11), and especially hidden Markov trees [Fan and Xia, 2003] are designed directly considering the intrinsic properties of wavelet transforms and combine the multiscale approach offered by wavelets with modeling of mutual statistical dependencies and non-Gaussian statistics frequently encountered in real-world texture analysis [Crouse et al., 1998].

15.1.9 Other statistical methods of texture description

there is a rich variety of other texture description techniques; we present here only the basic principles of the chief ones [Haralick, 1979; Ahuja and Rosenfeld, 1981; Davis et al., 1983; Derin and Elliot, 1987; Tomita and Tsuji, 1990].

The **mathematical morphology** approach looks for spatial repetitiveness of shapes in a binary image using structure primitives (see Chapter 13). If the structuring elements consist of a single pixel only, the resulting description is an autocorrelation function of the binary image. Using larger and more complex structuring elements, general correlation can be calculated—the element usually represents some simple shape, such as a square, a line, etc. When a binary textured image is eroded texture properties are present in the eroded image [Serra and Verchery, 1973]. One possibility for feature vector construction is to apply different structuring elements to the textured image and to count the number of pixels with unit value in the eroded images, each number forming one element of the feature vector. The morphological approach stresses the shape properties of texture primitives, but its applicability is limited due to the assumption of a binary textured image: gray-level mathematical morphology may help to solve this problem. Morphological texture description is often successful in granulated materials, which can be segmented by thresholding. Using a sequence of openings and counting the number of pixels after each step, a texture measure was derived in [Dougherty et al., 1989].

The **texture transform** represents another approach. Each texture type present in an image is transformed into a unique gray-level; the general idea is to construct an image g where the pixels $g(i, j)$ describe a texture in some neighborhood of the pixel $f(i, j)$ in the original textured image f . If micro-textures are analyzed, a small neighborhood of $f(i, j)$ must be used, and an appropriately larger neighborhood should be used for description of macro-textures. In addition, a priori knowledge can be used to guide the transformation and subsequent texture recognition and segmentation. Local texture

orientation can also be used to transform a texture image into a feature image, after which supervised classification is applied to recognize textures.

Linear estimates of gray-levels in texture pixels can also be used for texture description. Pixel gray-levels are estimated from gray-levels in their neighborhood—this method is based on the **autoregression texture model**, where linear estimation parameters are used [Deguchi and Morishita, 1978]. Model parameters vary substantially in fine textures, but remain relatively unchanged in coarse textures. This model has been compared with second-order spatial statistics [Gagalowicz et al., 1988]; it was found that although the results were comparable, spatial statistics performed more quickly and reliably.

The **peak and valley** method [Mitchell et al., 1977; Ehrick and Foith, 1978] is based on detection of local extrema of the brightness function in vertical and horizontal scans of a texture image. Fine textures have a large number of small-sized local extrema, coarse textures are represented by a smaller number of larger-sized local extrema—higher peaks and deeper valleys.

The sequence of pixel gray-levels can be considered a **Markov chain** in which transition probabilities of an m^{th} -order chain represent $(m + 1)^{th}$ -order statistics of textures [Pratt and Faugeras, 1978]. This approach may also be used for texture synthesis [Gagalowicz, 1979].

A more recent development is the **deterministic tourist walk** which uses a partially self-avoiding deterministic walk for texture analysis [Backes et al., 2010]. The method is inherently multi-scale and demonstrated good performance in comparison with Gabor and Fourier transforms, and co-occurrence matrix methods. Another scale-invariant approach using a novel family of texture description filters invariant to orientation, contrast, and scale and robust to local skew has been demonstrated [Mellor et al., 2008]. Successful application for image retrieval was based on χ -squared similarity measures applied to histograms built from local filter responses.

Many of the texture description features presented so far are interrelated; the Fourier power spectrum, the autoregression model, and autocorrelation functions represent the same subset of second-order statistics. The mathematical relationships between texture description methods are summarized in [Tomita and Tsuji, 1990], an experimental comparison of performance between several methods can be found in [Du Buf et al., 1990; Iversen and Lonnestad, 1994; Zhu and Goutte, 1995; Wang et al., 1996], and criteria for comparison are discussed in [Soh et al., 1991].

It has been shown that higher than second-order statistics contain little information that can be used for texture discrimination [Julesz and Caelli, 1979]. Nevertheless, identical second-order statistics do not guarantee identical textures; examples can be found in [Julesz and Bergen, 1987] together with a study of human texture perception. Texture-related research of the human visual system seems to bring useful results, and a texture analysis method based on studies of it was designed to emulate the process of texture feature extraction in each individual channel in the multi-channel spatial filtering model of perception [Rao, 1993].

15.2 Syntactic texture description methods

Syntactic and hybrid texture description methods are not as widely used as statistical approaches [Tomita et al., 1982]. **Syntactic** texture description is based on an analogy between the texture primitive spatial relations and the structure of a formal language.

Descriptions of textures from one class form a language that can be represented by its grammar, which is inferred from a training set of words of the language (from descriptions of textures in a training set)—during a learning phase, one grammar is constructed for each texture class present in the training set. The recognition process is then a syntactic analysis of the texture description word. The grammar that can be used to complete the syntactic analysis of the description word determines the texture class (see Section 9.4).

Purely syntactic texture description models are based on the idea that textures consist of primitives located in almost regular relationships. Primitive descriptions (discussed at the beginning of this chapter) and rules of primitive placement must be determined to describe a texture [Tsuji and Tomita, 1973; Lu and Fu, 1978]. One of the most efficient ways to describe the structure of primitive relationships is using a grammar which represents a rule for building a texture from primitives, by applying transformation rules to a limited set of symbols. Symbols represent the texture primitive types and transformation rules represent the spatial relations between primitives. In Chapter 9 it was noted that any grammar is a very strict formula, but textures of the real world are usually irregular, and structural errors, distortions, or even structural variations are frequent. This means that no strict rule can be used to describe a texture in reality. To make syntactic description of real textures possible, variable rules must be incorporated into the description grammars, and non-deterministic or stochastic grammars must be used (see Section 9.4 and [Fu, 1974]). Further, there is usually no single description grammar for a texture class, which might be described by an infinite number of different grammars using different symbols and different transformation rules, and different grammar types as well. Here, we will discuss chain grammars and graph grammars; other grammars suitable for texture description (tree, matrix) can be found in [Ballard and Brown, 1982; Fu, 1982; Vafaie and Bourbakis, 1988]. Another approach to texture description using generative principles is to use **fractals** [Mandelbrot, 1982; Barnsley, 1988].

15.2.1 Shape chain grammars

The simplest grammars that can be used for texture description are shape chain grammars (see Section 9.4). They generate textures beginning with a start symbol followed by application of transform rules, called **shape rules**—the process ends if no further transform rule can be applied. Texture synthesis consists of two steps: first, the transform rule is found, and then the rule must be geometrically adjusted to match the generated texture exactly (rules are more general; they may not include size, orientation, etc.).

Algorithm 15.5: Shape chain grammar texture synthesis

1. Start a synthesis process by applying some transform rule to the start symbol.
2. Find a part of a previously generated texture that matches the left side of some transform rule. This match must be an unambiguous correspondence between terminal and non-terminal symbols of the left-hand side of the chosen transform rule with terminal and non-terminal symbols of the part of the texture to which the rule is applied. If no such part of the texture can be found, stop.
3. Find an appropriate geometric transform that can be applied to the left-hand side of the chosen rule to match it to the considered texture part exactly.

4. Apply this transform to the right-hand side of the transform rule.
5. Substitute the specified part of the texture (the part that matches a geometrically transformed left-hand side of the chosen rule) with the geometrically transformed right-hand side of the chosen transform rule.
6. Continue with step 2.

We can demonstrate this algorithm on an example of hexagonal texture synthesis. Let V_n be a set of non-terminal symbols, V_t a set of terminal symbols, R a set of rules, S the start symbol (as in Section 9.4). The grammar [Ballard and Brown, 1982] is illustrated in Figure 15.11, which can then be used to generate hexagonal texture following Algorithm 15.5—note that the non-terminal symbol may appear in different rotations. Rotation of primitives here is represented by a small circle attached to one side of the primitive hexagon in Figure 15.11. Recognition of hexagonal textures is the proof that the texture can be generated by this grammar; the texture recognition uses syntactic analysis as described in Section 9.4. Note that according to the grammar of Figure 15.11, the texture shown in Figure 15.12a will be accepted, but that of Figure 15.12b will be rejected.

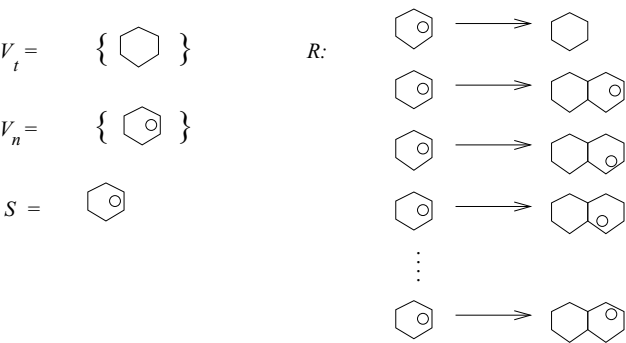


Figure 15.11: Grammar generating hexagonal textures. © Cengage Learning 2015.

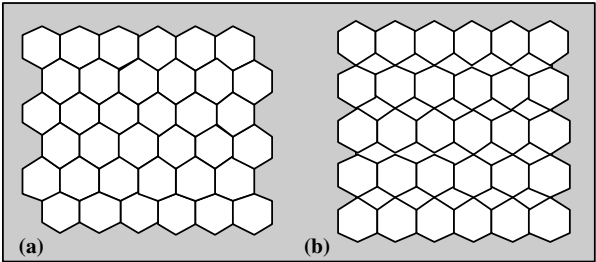


Figure 15.12: Hexagonal textures. (a) Accepted; (b) Rejected. © Cengage Learning 2015.

15.2.2 Graph grammars

Texture analysis is more common than texture synthesis in machine vision tasks (even if texture synthesis is generally more common, e.g., in computer graphics and computer games). The natural approach to texture recognition is to construct a planar graph of primitive layout and to use it in the recognition process. Primitive classes and primitive spatial relations must be known to construct such a graph; spatial relationships between texture primitives will then be reflected in the graph structure. Texture primitive classes will be coded in graph nodes, each primitive having a corresponding node in the graph, and two nodes will be connected by an arc if there is no other primitive in some specified neighborhood of these two primitives. The size of this neighborhood is the main influence on the complexity of the resulting planar graph—the larger the neighborhood, the smaller the number of arcs. Note that choosing the neighborhood too large may result in no arcs for some nodes (the same may be true for the neighborhood being too small). Characteristic properties of some graphs used practically (relative neighborhood graphs, Gabriel graphs, Voronoi diagrams) are described in [Urquhart, 1982; Ahuja, 1982; Tuceryan and Jain, 1990]. These graphs are undirected since the spatial neighborhood relation is symmetric, with weighted arcs and nodes. Each node is labeled with a primitive class to which it corresponds, and arcs are weighted by their length and direction.

The texture classification problem is then transformed into a graph recognition problem for which the following approaches may be used.

1. Simplify the texture description by decomposition of the planar graph into a set of chains (sequences of adjacent nodes), and apply the algorithms discussed in the previous section. The chain descriptions of textures can represent border primitives of closed regions, different graph paths, primitive neighborhood, etc. A training set is constructed from the decomposition of several texture description planar graphs for each texture class. Appropriate grammars are inferred which represent textures in the training sets. The presence of information noise is highly probable, so stochastic grammars should be used. Texture classification consists of the following steps.
 - A classified texture is represented by a planar graph.
 - The graph is decomposed into chains.
 - The description chains are presented for syntactic analysis.
 - A texture is classified into the class whose grammar accepts all the chains of the decomposed planar graph. If more than one grammar accepts the chains, the texture can be classified into the class whose grammar accepted the chains with the highest probability.

The main advantage of this approach is its simplicity but a disadvantage is the impossibility of reconstructing the original planar graph from the chain decomposition—some portion of the syntactic information is lost during decomposition.

2. Another class of planar graph description may be represented by a stochastic graph grammar or by an extended graph grammar for description of distorted textures. This approach is very difficult from both the implementational and algorithmic points of view; the main problem is in grammar inference.

3. The planar graphs can be compared directly using graph matching approaches. It is necessary to define a 'distance' between two graphs as a measure of their similarity; if such a distance is defined, standard methods used in statistical classifier learning can be used—exemplar computation, cluster analysis, etc.

The syntactic approach is valued for its ability to describe a texture character at several hierarchical levels. It permits a qualitative analysis of textures, for decomposition into descriptive substructures (primitive grouping), to incorporate texture descriptions into the whole description of image, scene, etc. From this point of view, it significantly exceeds the complexity of simple object classification. Not considering the implementation difficulties, the second approach from the list above is recommended; if a descriptive graph grammar is chosen appropriately, it can generate a class of graphs independently of their size. It can be used if a pattern is sought in an image at any hierarchical level. An example of a planar graph describing a texture is shown in Figure 15.13.

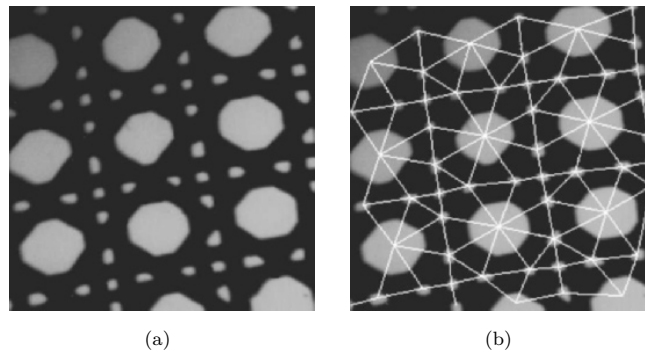


Figure 15.13: Planar graph describing a texture. (a) Texture primitives. (b) Planar graph overlaid. © Cengage Learning 2015.

15.2.3 Primitive grouping in hierarchical textures

Several levels of primitives can be detected in hierarchical textures—lower-level primitives form some specific pattern which can be considered a primitive at a higher description level (Figure 15.14). The process of detecting these primitive patterns (units) in a texture is called **primitive grouping**. Note that these units may form new patterns at an even higher description level. Therefore, the grouping process must be repeated until no new units can be formed.

Grouping makes a syntactic approach to texture segmentation possible, and plays the same role as local computation of texture features in statistical texture recognition. It has been claimed several times that different primitives and/or different spatial relationships represent different textures. Consider an example (Figure 15.15a) in which the primitives are the same (dots) and textures differ in the spatial relations between them. If a higher hierarchical level is considered, different primitives can be detected in both textures—the textures do not consist of the same primitive types any more (Figure 15.15b).

A primitive grouping algorithm is described in [Tomita and Tsuji, 1990].

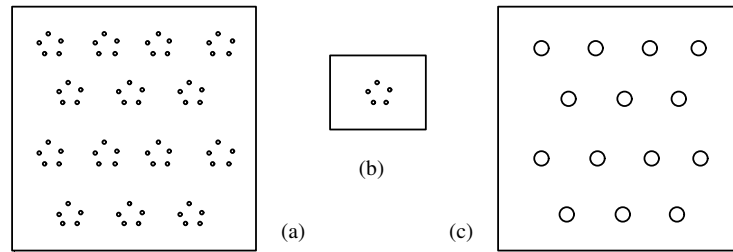


Figure 15.14: Hierarchical texture. (a) Texture. (b) A pattern formed from low-level primitives, this pattern can be considered a primitive in the higher level. (c) Higher-level texture. © Cengage Learning 2015.

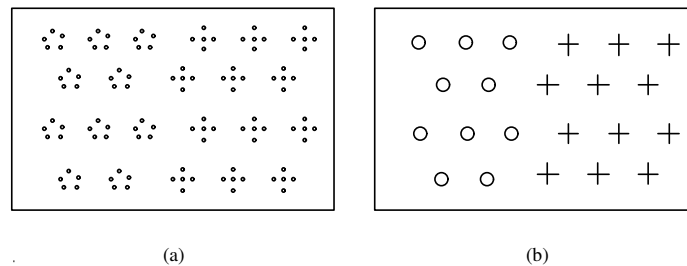


Figure 15.15: Primitive grouping. (a) Two textures, same primitives in the lowest description level. (b) The same two textures, different primitives in the higher description level. © Cengage Learning 2015.

Algorithm 15.6: Texture primitive grouping

1. Determine texture primitive properties and classify primitives into classes.
2. Find the nearest and the second nearest neighbor for each texture primitive. Using the primitive class and distances to the nearest two neighboring primitives d_1, d_2 , classify low-level primitives into **new** classes, see Figure 15.16.
3. Primitives with the same **new** classification which are connected (close to each other), are linked together and form higher-level primitives, see Figure 15.16.
4. If any two resulting homogeneous regions of linked primitives overlap, let the overlapped part form a separate region, see Figure 15.17.

Regions formed from primitives of the lower level may be considered primitives in the higher level and the grouping process may be repeated for these new primitives. Nevertheless, sophisticated control of the grouping process is necessary to achieve meaningful results—it must be controlled from a high-level vision texture understanding subsystem. A recursive primitive grouping, which uses histograms of primitive properties and primitive spatial relations is presented in [Tomita and Tsuji, 1990] together with examples of syntactic-based texture segmentation results.

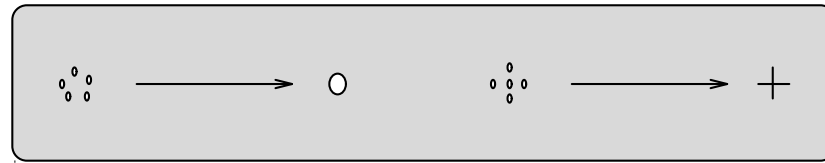


Figure 15.16: Primitive grouping—low-level primitive patterns are grouped into single primitives at a higher level. © Cengage Learning 2015.

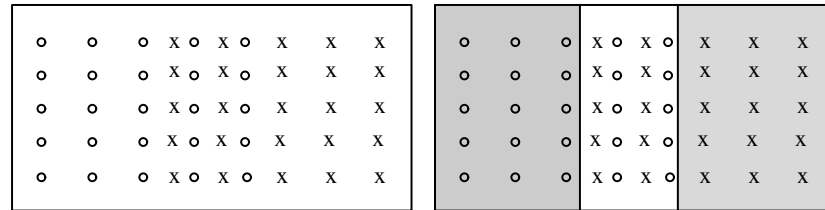


Figure 15.17: Overlap of homogeneous regions results in their splitting. © Cengage Learning 2015.

15.3 Hybrid texture description methods

Purely syntactic methods of texture description experience many difficulties with syntactic analyzer learning and with graph (or other complex) grammar inference, which is why purely they are not widely used. On the other hand, a precise definition of primitives brings many advantages and it is not wise to avoid it completely. Hybrid methods of texture description combine the statistical and syntactic approaches; they are partly syntactic because primitives are exactly defined, and partly statistical because spatial relations between primitives are based on probabilities [Conners and Harlow, 1980].

The hybrid approach to texture description distinguishes between weak and strong textures. The syntactic part of weak texture description divides an image into regions based on a tonal image property (e.g., constant gray-level regions) which are considered texture primitives. Primitives can be described by their shape, size, etc. The next step constructs histograms of sizes and shapes of all the texture primitives contained in the image. If the image can be segmented into two or more sets of homogeneous texture regions, the histogram is bi-modal and each primitive is typical of one texture pattern. This can be used for texture segmentation.

If the starting histogram does not have significant peaks, a complete segmentation cannot be achieved. The histogram-based segmentation can be repeated in each hitherto segmented homogeneous texture region. If any texture region consists of more than one primitive type, the method cannot be used and spatial relations between primitives must be computed. Some methods are discussed in [Haralick, 1979].

Description of strong textures is based on the spatial relations of texture primitives and two-directional interactions between primitives seem to carry most of the information. The simplest texture primitive is a pixel and its gray-level property, while the maximum contiguous set of pixels of constant gray-level is a more complicated texture primitive [Wang and Rosenfeld, 1981]. Such a primitive can be described by its size, elongatedness, orientation, average gray-level, etc. The texture description includes spa-

tial relations between primitives based on distance and adjacency relations. Using more complex texture primitives brings more textural information. On the other hand, all the properties of single pixel primitives are immediately available without the necessity of being involved in extensive primitive property computations.

The hybrid multi-level texture description and classification method [Sonka, 1986] is based on primitive definition and spatial description of inter-primitive relations. The method considers both tone and structural properties and consists of several steps. Texture primitives are extracted first, and then described and classified. As a result of this stage, a classifier knows how to classify texture primitives. Known textures are then presented to the texture recognition system. Texture primitives are extracted from the image and the first-level classifier recognizes their classes. Based on recognized texture primitives, spatial relations between primitive classes are evaluated for each texture from the training set. Spatial relations between texture primitives are described by a feature vector used to adjust a second-level classifier. If the second-level classifier is set, the two-level learning process is over, and unknown textures can be presented to the texture recognition system. The primitives are classified by the first-level classifier, spatial primitive properties are computed and the second-level classifier assigns the texture to one of the texture classes. Some hybrid methods use Fourier descriptors for shape coding and a texture is modeled by a reduced set of joint probability distributions obtained by vector quantization. The LBP approach presented earlier in Section 15.1.6 is an inherently hybrid texture description approach—uniform patterns can be understood as texture primitives while estimating the distributions of these primitives (microtemplates) combines the structural and statistical image analysis [Ojala et al., 2002b].

15.4 Texture recognition method applications

Textures are very common in our world, and application possibilities are almost unlimited: estimated crop yields and localization of diseased forests from remotely sensed data, automatic diagnosis of lung diseases from X-ray images, recognition of cloud types from meteorological satellite data, detection of textile pattern imperfections, are just some examples of applications.

Classical applications of spatial frequency-based texture description methods include recognition of roads, road crossings, buildings, agricultural regions, and natural objects, or classification of trees into classes. An interesting proof of the role of textural information in outdoor object recognition was done by comparison of classification correctness if textural information was and was not used. Spectral information-based classification achieved 74% correctly classified objects, while adding textural information increased accuracy to 99% [Haralick, 1979]. Industrial applications of texture description and recognition are becoming more and more common. Examples can be found in almost all branches of industrial and biomedical activities—quality inspection in the motor or textile industries [Wood, 1990; Xie, 2008], workpiece surface monitoring, road surface skidding estimation, micro-electronics, remote sensing, mammography [Miller and Astley, 1992], MR brain imaging [Toulson and Boyce, 1992], pulmonary parenchyma characterization [Xu et al., 2006], three-dimensional texture analysis [Ip and Lam, 1995; Kovalev et al., 2001], content-based data retrieval from image databases, etc.

15.5 Summary

- **Texture**

- Texture is widely used and intuitively obvious but has no precise definition due to its wide variability.
- Texture consists of texture **primitives** (texture **elements**) called **texels**.
- A texture primitive is a contiguous set of pixels with some tonal and/or regional property.
- **Texture description** is based on **tone** and **structure**. Tone describes pixel intensity properties in the primitive, while structure reflects spatial relationships between primitives.
- Texture description is **scale dependent**.
- **Statistical** methods of texture description compute different texture properties and are suitable if texture primitive sizes are comparable with the pixel sizes.
- **Syntactic** and **hybrid** methods (combination of statistical and syntactic) are more suitable for textures where primitives can be easily determined and their properties described.

- **Statistical texture description**

- Statistical methods describe textures in a form suitable for statistical pattern recognition. Each texture is described by a feature vector of properties which represents a point in a multi-dimensional feature space.
- Coarse textures are built from larger primitives, fine textures from smaller primitives. Textural character is in direct relation to the spatial size of the primitives.
- Fine textures are characterized by higher spatial frequencies, coarse textures by lower spatial frequencies.
- Measuring spatial frequencies is the basis of a large group of texture recognition methods:
 - * Autocorrelation function of a texture
 - * Optical image transform
 - * Discrete image transform
- Texture description may be based on repeated occurrence of some gray-level configuration in the texture; this configuration varies rapidly with distance in fine textures, slowly in coarse textures. **Co-occurrence matrices** represent such an approach.
- **Edge frequency** approaches describe frequencies of edge appearances in texture.
- In the **primitive length (run length)** approach, texture description features can be computed as continuous probabilities of the length and the gray-level of primitives in the texture.

- **Laws' texture measures** determine texture properties by assessing average gray-level, edges, spots, ripples, and waves in texture.
 - **Local Binary Patterns (LBP)** identify local microstructure templates and estimate their distribution in the texture. Local neighborhoods are thresholded to binary patterns, histogram distributions of which are used for description.
 - The **fractal** approach to texture description is based on correlation between texture coarseness and fractal dimension and texture granularity and lacunarity.
 - **Wavelet** texture description
 - * These are typically more efficient than other statistical methods.
 - * Wavelet energy signatures or their second-order statistics are frequently used.
 - * Standard wavelet approaches are not translation invariant.
 - * Discrete wavelet frames introduce translational invariance and can be efficiently implemented using filter banks.
 - * Wavelet-based hidden Markov models and hidden Markov trees incorporate independence between wavelet subbands for additional performance enhancement.
 - Other statistical approaches exist:
 - * Mathematical morphology
 - * Texture transform
 - Varieties of texture properties may be derived from **first-order** and **second-order statistics** of elementary measures such as co-occurrences, edge distributions, primitive lengths, etc.
 - Higher order statistics contain little information that can be used for texture discrimination.
- **Syntactic and hybrid texture description**
 - Syntactic texture description is based on an analogy between texture primitive spatial relations and structure of a formal language.
 - Hybrid methods combine statistical and syntactic approaches; they are partly syntactic because primitives are exactly defined, and partly statistical because spatial relations between primitives are based on probabilities.
 - Purely syntactic texture description models utilize the idea that textures consist of primitives located in almost regular relationships. Primitive descriptions and rules of primitive placement must be determined to describe a texture.
 - Textures of the real world are usually irregular, with frequent structural errors, distortions, and/or structural variations causing strict grammars to be inapplicable. To make syntactic description of real textures possible, variable rules must be incorporated into the description grammars, and non-deterministic or stochastic grammars must be used.

- Syntactic texture description methods include:
 - * **Shape chain grammars**, which are the simplest grammars that can be used for texture description. They generate textures beginning with a start symbol followed by application of **shape transform rules**.
 - * **Graph grammars** construct a **planar graph of primitive layout**. Primitive classes and their spatial relations must be known to construct such a graph; spatial relationships between texture primitives are reflected in the graph structure. Texture classification is then transformed into a **graph recognition** problem.
- The syntactic approach is valued for its ability to describe a texture character at several **hierarchical levels**.
- **Primitive grouping** can be performed if lower-level primitives that form some specific pattern can be considered a primitive at a higher description level.
- Syntactic and hybrid texture description methods are not as widely used as statistical approaches.

15.6 Exercises

Short-answer questions

- S15.1 What is a texel?
- S15.2 Explain the difference between weak and strong textures.
- S15.3 Explain the difference between fine and coarse textures.
- S15.4 What is the role of scale in texture description?
- S15.5 Specify the main texture description and recognition strategies employed in statistical, syntactic, and hybrid methods. For each of the three general approaches, describe the texture types for which the approach can be expected to perform well and for which it is not appropriate.
- S15.6 Determine whether the following texture features exhibit relatively higher or lower values in fine textures compared to coarse textures:
- (a) Energy in a small-radius circle of the Fourier power spectrum (Figure 15.3a)
 - (b) Average edge frequency feature (equation 15.10) for a small value of d
 - (c) Short primitive emphasis
 - (d) Long primitive emphasis
 - (e) Fractal dimension
 - (f) Lacunarity
- S15.7 Name several texture description features that are well suited for characterization of directional textures.
- S15.8 Explain why deterministic grammars are too restrictive for description of real-world textures.
- S15.9 On a rectangular grid, sketch the geometric pattern of an $LBP_{4,4}$ texture description feature detector.

- S15.10 What is the main reason for rotational invariance of $\text{LBP}_{P,R}^{r,i}$ texture descriptor compared to $\text{LBP}_{P,R}$?
- S15.11 Explain why use of LTPs as opposed to LBPs for texture description compromises gray-level transform invariance.
- S15.12 Define fractal dimension and lacunarity, explain how these measures can be used for texture description.
- S15.13 Define a stochastic grammar. How is it useful for texture description?
- S15.14 Explain the rationale for primitive grouping. What are its advantages? To which textures is this approach applicable? How can it be used for texture description?
- S15.15 Suggest some application areas in which texture description and recognition can be used.

Problems

- P15.1 Using the World Wide Web, find several images of dissimilar homogeneous textures (Brodatz textures [Brodatz, 1966] from a Web-based database may be a good choice)
1. Create your personal database TD1 from these images of at least 5 texture type ranging gradually from fine to coarse.
 2. Create a texture database TD2 of three dissimilar texture classes with at least ten images belonging to each class.
 3. Create a database TD3 of at least three homogeneous directional textures (use preferably several images from each class) and rotate each at 9 random angles – this will form a database of 10 directional images for each texture.

These three databases will be used in some of the experiments below.

- P15.2 Implement Fourier transform based descriptors.
- P15.3 Determine the co-occurrence matrices $P_{0^\circ,2}$, $P_{45^\circ,2}$, and $P_{90^\circ,3}$ for the image in Figure 15.4.
- P15.4 Determine the average edge frequency function (equation 15.10) for $d \in \{1, 2, 3, 4, 5\}$ for a 30×30 checkerboard image with 3×3 binary checkers (values of 0 and 1, 0-level checker in the upper left corner).
- P15.5 For a 30×30 checkerboard image with 3×3 binary checkers (values of 0 and 1, 0-level checker in the upper left corner) and a 30×30 image with vertical binary 3-pixel-wide stripes (0-level stripe along the left image edge), determine the following non-directional texture descriptors:
- (a) Short primitives emphasis
 - (b) Long primitives emphasis
 - (c) Gray-level uniformity
 - (d) Primitive length uniformity
 - (e) Primitive percentage
- P15.6 Repeat Problem P15.5 with modified texture descriptors that consider directionality of primitives.
- P15.7 Develop functions for computing the following texture features in an image of a given size:

- (a) Co-occurrence descriptors
 - (b) Average edge frequency
 - (c) Primitive length descriptors
 - (d) Laws' energy descriptors
 - (e) $LBP_{P,R}^{riu2}$ descriptors
 - (f) Fractal texture descriptors
- P15.8 For five highly dissimilar textures ranging from fine to coarse selected from your database TD1 (Problem P15.1) and using the functions developed in Problem P15.7, determine:
- (a) Co-occurrence features specified by equations (15.4–15.9)
 - (b) Average edge frequency
- for four distinct values of d (e.g., $d = 1, 3, 5, 7$; note that you may select other values of d according to the scale of your textures). Select three texture descriptors to determine what can be concluded from inspection of the relationship between the texture descriptors and the value of d in each texture. Further, what can be concluded for the relationship between texture coarseness and the values of individual texture descriptors?
- P15.9 For five highly dissimilar textures ranging from fine to coarse selected from your database TD1 (Problem P15.1) and using the functions developed in Problem P15.7, determine:
- (a) Primitive length features specified by equations (15.12)–(15.16)
 - (b) Laws' energy features
 - (c) $LBP_{P,R}^{riu2}$ features
 - (d) Fractal-based texture features—fractal dimension and lacunarity
- What can be concluded for the relationship between texture coarseness and the values of individual texture descriptors?
- P15.10 Design a shape chain grammar that generates the texture shown in Figure 12.4. Show the first few steps of the generation process.
- P15.11 Design a shape chain grammar that generates the texture shown in Figure 15.12b.
- P15.12 Using your database of classes TD2 (Problem P15.1), develop a program for statistical texture description and recognition. Texture descriptors should reflect the discriminative properties of the three classes. Train a simple statistical classifier (Chapter 9) using half of the texture images from each class, and determine classification correctness using the remaining texture images.
- P15.13 Using your database TD3 (Problem P15.1) of texture classes that include rotated images, develop a program for statistical texture description and recognition so that textures are recognized regardless of their orientation. Texture descriptors should reflect the discriminative properties of the three classes. Do not use LBP descriptors for this problem. Train a simple statistical classifier (Chapter 9) using half of the texture images from each class, and determine classification correctness using the remaining texture images.
- P15.14 Repeat Problem P15.13 using LBP and LBP^{riu2} texture descriptors. Compare the performance of these two descriptors with each other as well as with results obtained in Problem P15.13

- P15.15 Using the images from your databases TD1–TD3 (Problem P15.1) create several artificial images that contain three to five regions of different texture. Develop a texture classification program that performs texture recognition in rectangular windows. Using a moving window, the program should classify each window into one of the classes. Train the classifier in single-texture images using windows of the same size as is used in the classification phase. Assess image segmentation accuracy as the percentage of correctly classified image area.
- P15.16 Make yourself familiar with solved problems and **Matlab** implementations of selected algorithms provided in the corresponding chapter of the **Matlab Companion** to this text [Svoboda et al., 2008]. The **Matlab Companion** homepage <http://visionbook.felk.cvut.cz> offers images used in the problems, and well-commented **Matlab** code is provided for educational purposes.
- P15.17 Use the **Matlab Companion** [Svoboda et al., 2008] to develop solutions to additional exercises and practical problems provided there. Implement your solutions using **Matlab** or other suitable programming languages.

15.7 References

- Ahuja N. Dot pattern processing using Voronoi neighborhood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:336–343, 1982.
- Ahuja N. and Rosenfeld A. Mosaic models for textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(1):1–11, 1981.
- Arivazhagan S. and Ganesan L. Texture classification using wavelet transform. *Pattern Recogn. Lett.*, 24:1513–1521, 2003.
- Backes A. R., Goncalves W. N., Martinez A. S., and Bruno O. M. Texture analysis and classification using deterministic tourist walk. *Pattern Recognition*, 43:686–694, 2010.
- Ballard D. H. and Brown C. M. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- Barnsley M. F. *Fractals Everywhere*. Academic Press, Boston, 1988.
- Besicovitch A. S. and Ursell H. D. Sets of fractional dimensions (V): On dimensional numbers of some continuous curves. *Journal of the London Mathematical Society*, 12:18–25, 1937.
- Bovik A. C., Clark M., and Geisler W. S. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:55–73, 1990.
- Brodatz P. *Textures: A Photographic Album for Artists and Designers*. Dover, Toronto, 1966.
- Castleman K. R. *Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1996.
- Chang T. and Kuo C. C. Texture analysis and classification with tree-structure wavelet transform. *IEEE Transactions on Image Processing*, 2:429–441, 1993.
- Coggins J. M. and Jain A. K. A spatial filtering approach to texture analysis. *Pattern Recognition Letters*, 3:195–203, 1985.
- Connors R. W. and Harlow C. A. Toward a structural textural analyser based on statistical methods. *Computer Graphics and Image Processing*, 12:224–256, 1980.
- Crouse M. S., Nowak R. D., and Baraniuk R. G. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46:886–902, 1998.
- Davis L. S. and Mitiche A. Edge detection in textures. *Computer Graphics and Image Processing*, 12:25–39, 1980.
- Davis L. S., Janos L., and Dunn S. M. Efficient recovery of shape from texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(5):485–492, 1983.

- Deguchi K. and Morishita I. Texture characterization and texture-based partitioning using two-dimensional linear estimation. *IEEE Transactions on Computers*, 27:739–745, 1978.
- Derin H. and Elliot H. Modelling and segmentation of noisy and textured images using Gibbs random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1): 39–55, 1987.
- Dougherty E. R., Kraus E. J., and Pelz J. B. Image segmentation by local morphological granulometries. In *Proceedings of IGARSS '89 and Canadian Symposium on Remote Sensing*, Vancouver, Canada, pages 1220–1223, New York, 1989. IEEE.
- Du Buf J. M. H., Kardan M., and Spann M. Texture feature performance for image segmentation. *Pattern Recognition*, 23(3–4):291–309, 1990.
- Ehrick R. W. and Foith J. P. A view of texture topology and texture description. *Computer Graphics and Image Processing*, 8:174–202, 1978.
- Fan G. and Xia X. G. Wavelet-based texture analysis and synthesis using hidden Markov models. *IEEE Trans. Circuits and Systems*, 50:106–120, 2003.
- Fan Z. Edge-based hierarchical algorithm for textured image segmentation. In *International Conference on Acoustics, Speech, and Signal Processing*, Glasgow, Scotland, pages 1679–1682, Piscataway, NJ, 1989. IEEE.
- Fu K. S. *Syntactic Methods in Pattern Recognition*. Academic Press, New York, 1974.
- Fu K. S. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- Fung P. W., Grebbin G., and Attikiouzel Y. Contextual classification and segmentation of textured images. In *Proceedings of the 1990 International Conference on Acoustics, Speech, and Signal Processing—ICASSP 90*, Albuquerque, NM, pages 2329–2332, Piscataway, NJ, 1990. IEEE.
- Gagalowicz A. Stochastic texture fields synthesis from a priori given second order statistics. In *Proceedings, Pattern Recognition and Image Processing*, Chicago, IL, pages 376–381, Piscataway, NJ, 1979. IEEE.
- Gagalowicz A., Graffigne C., and Picard D. Texture boundary positioning. In *Proceedings of the 1988 IEEE International Conference on Systems, Man, and Cybernetics*, pages 16–19, Beijing/Shenyang, China, 1988. IEEE.
- Galloway M. M. Texture classification using gray level run length. *Computer Graphics and Image Processing*, 4:172–179, 1975.
- Gotlieb C. C. and Kreyszig H. E. Texture descriptors based on co-occurrence matrices. *Computer Vision, Graphics, and Image Processing*, 51(1):70–86, 1990.
- Haralick R. M. Statistical and structural approaches to texture. *Proceedings IEEE*, 67(5): 786–804, 1979.
- Haralick R. M., Shanmugam K., and Dinstein I. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3:610–621, 1973.
- Hausdorff F. Dimension und ausseres Mass. *Mathematische Annalen*, 79:157–179, 1919.
- Huang Q., Lorch J. R., and Dubes R. C. Can the fractal dimension of images be measured? *Pattern Recognition*, 27:339–349, 1994.
- Hurst H. E. Long-term storage capacity of reservoirs. *Transactions of the American Society of Civil Engineers*, 116:770–808, 1951.
- Ip H. H. S. and Lam S. W. C. Three-dimensional structural texture modeling and segmentation. *Pattern Recognition*, 28:1299–1319, 1995.

- Iversen H. and Lonnestad T. An evaluation of stochastic models for analysis and synthesis of gray-scale texture. *Pattern Recognition Letters*, 15:575–585, 1994.
- Jin X. C., Ong S. H., and Jayasooriah. A practical method for estimating fractal dimension. *Pattern Recognition Letters*, 16:457–464, 1995.
- Julesz B. Textons, the elements of texture perception, and their interactions. *Nature*, 290: 91–97, 1981.
- Julesz B. and Bergen J. R. Textons, the fundamental elements in preattentive vision and perception of textures. In *Readings in Computer Vision*, pages 243–256. Morgan Kaufmann Publishers, Los Altos, CA, 1987.
- Julesz B. and Caelli T. On the limits of Fourier decompositions in visual texture perception. *Perception*, 8:69–73, 1979.
- Keller J. M., Chen S., and Crownover R. M. Texture description and segmentation through fractal geometry. *Computer Vision, Graphics, and Image Processing*, 45(2):150–166, 1989.
- Kjell B. P. and Wang P. Y. Noise-tolerant texture classification and image segmentation. In *Intelligent Robots and Computer Vision IX: Algorithms and Techniques*, Boston, pages 553–560, Bellingham, WA, 1991. SPIE.
- Kovalev V. A., Kruggel F., Gertz H. J., and Cramon D. Y. v. Three-dimensional texture analysis of MRI brain datasets. *IEEE Transactions on Medical Imaging*, 20:424–433, 2001.
- Laws K. I. Texture energy measures. In *DARPA Image Understanding Workshop*, Los Angeles, CA, pages 47–51, Los Altos, CA, 1979. DARPA.
- Liu S. S. and Jernigan M. E. Texture analysis and discrimination in additive noise. *Computer Vision, Graphics, and Image Processing*, 49:52–67, 1990.
- Lu S. Y. and Fu K. S. A syntactic approach to texture analysis. *Computer Graphics and Image Processing*, 7:303–330, 1978.
- Lundahl T., Ohley W. J., Kay S. M., and Siffert R. Fractional Brownian motion: A maximum likelihood estimator and its application to image texture. *IEEE Transactions on Medical Imaging*, 5:152–161, 1986.
- Mallat S. G. A theory of multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.
- Mandelbrot B. B. *The Fractal Geometry of Nature*. Freeman, New York, 1982.
- Mellor M., Hong B. W., and Brady M. Locally rotation, contrast, and scale invariant descriptors for texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30: 52–61, 2008.
- Miller P. and Astley S. Classification of breast tissue by texture analysis. *Image and Vision Computing*, 10(5):277–282, 1992.
- Mitchell O. R., Myer C. R., and Boyne W. A max-min measure for image texture analysis. *IEEE Transactions on Computers*, 26:408–414, 1977.
- Mojilovic A., Popovic M. V., and Rackov D. M. On the selection of an optimal wavelet basis for texture characterization. *IEEE Transactions on Image Processing*, 9:2043–2050, 2000.
- Ojala T., Pietikainen M., and Harwood D. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29:51–59, 1996.
- Ojala T., Valkealahti K., Oja E., and Pietikainen M. Texture discrimination with multidimensional distributions of signed gray level differences. *Pattern Recognition*, 34:727–739, 2001.

- Ojala T., Maenpaa T., Pietikainen M., Viertola J., Kyllonen J., and Huovinen S. Outex - new framework for empirical evaluation of texture analysis algorithms. In *Proc. 16th International Conference on Pattern Recognition*, volume 1, pages 701–706, Quebec, Canada, 2002a.
- Ojala T., Pietikainen M., and Maenpaa M. Multiresolution gray-scale and rotation invariant texture classification with locally binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:971–987, 2002b.
- Oppenheim A. V., Schaffer R. W., and Buck J. R. *Discrete-Time Signal Processing*. Prentice Hall, New York, 2nd edition, 1999.
- Pentland A. P. Fractal-based description of natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:661–674, 1984.
- Perry A. and Lowe D. G. Segmentation of non-random textures using zero-crossings. In *1989 IEEE International Conference on Systems, Man, and Cybernetics*, Cambridge, MA, pages 1051–1054, Piscataway, NJ, 1989. IEEE.
- Pietikainen M., Ojala T., and Xu Z. Rotation-invariant texture classification using feature distributions. *Pattern Recognition*, 33:43–52, 2000.
- Porter R. and Canagarajah N. Robust rotation-invariant texture classification: Wavelet, Gabor filter, and GMRF based schemes. *IEE Proc. Vision, Image, Signal Processing*, 144:180–188, 1997.
- Pratt W. K. and Faugeras O. C. Development and evaluation of stochastic-based visual texture features. *IEEE Transactions on Systems, Man and Cybernetics*, 8:796–804, 1978.
- Rao A. R. Identifying high level features of texture perception. *CVGIP – Graphical Models and Image Processing*, 55:218–233, 1993.
- Reed T. R., Wechsler H., and Werman M. Texture segmentation using a diffusion region growing technique. *Pattern Recognition*, 23(9):953–960, 1990.
- Rioul O. A discrete-time multiresolution theory. *IEEE Trans on Signal Proc.*, 41:2591–2606, 1993.
- Rosenfeld A., editor. *Digital Picture Analysis*. Springer Verlag, Berlin, 1976.
- Sarkar N. and Chaudhuri B. B. An efficient differential box-counting approach to compute fractal dimension of image. *IEEE Transactions on Systems, Man and Cybernetics*, 24: 115–120, 1994.
- Serra J. and Verchery G. Mathematical morphology applied to fibre composite materials. *Film Science Technology*, 6:141–158, 1973.
- Shulman A. R. *Optical Data Processing*. Wiley, New York, 1970.
- Sklansky J. Image segmentation and feature extraction. *IEEE Transactions on Systems, Man and Cybernetics*, 8(4):237–247, 1978.
- Soh Y., Murthy S. N. J., and Huntsberger T. L. Development of criteria to compare model-based texture analysis methods. In *Intelligent Robots and Computer Vision IX: Algorithms and Techniques*, Boston, pages 561–573, Bellingham, WA, 1991. SPIE.
- Sonka M. A new texture recognition method. *Computers and Artificial Intelligence*, 5(4): 357–364, 1986.
- Svoboda T., Kybic J., and Hlavac V. *Image Processing, Analysis, and Machine Vision: A MATLAB Companion*. Thomson Engineering, 2008.
- Tan X. and Triggs B. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *Image Processing, IEEE Transactions on*, 19:1635–1650, 2010.

- Tomita F. and Tsuji S. *Computer Analysis of Visual Textures*. Kluwer, Norwell, MA, 1990.
- Tomita F., Shirai Y., and Tsuji S. Description of textures by a structural analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(2):183–191, 1982.
- Toulson D. L. and Boyce J. F. Segmentation of MR images using neural nets. *Image and Vision Computing*, 10(5):324–328, 1992.
- Tsuji S. and Tomita F. A structural analyser for a class of textures. *Computer Graphics and Image Processing*, 2:216–231, 1973.
- Tuceryan M. and Jain A. K. Texture segmentation using Voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):211–216, 1990.
- Unser M. Local linear transforms for texture measurements. *Signal Processing*, 11:61–79, 1986.
- Unser M. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4:1549–1560, 1995.
- Unser M. and Eden M. Multiresolution feature extraction and selection for texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):717–728, 1989.
- Urquhart R. Graph theoretical clustering based on limited neighbourhood sets. *Pattern Recognition*, 15(3):173–187, 1982.
- Vafaie H. and Bourbakis N. G. Tree grammar scheme for generation and recognition of simple texture paths in pictures. In *Third International Symposium on Intelligent Control 1988*, Arlington, VA, pages 201–206, Piscataway, NJ, 1988. IEEE.
- Van de Wouwer G., Scheunders P., and Van Dyck D. Statistical texture characterization from wavelet representations. *IEEE Transactions on Image Processing*, 8:592–598, 1999.
- Vautrot P., Bonnet N., and Herbin M. Comparative study of different spatial/spatial frequency methods for texture segmentation/classification. In *Proceedings of the IEEE International Conference on Image Processing*, Lausanne, Switzerland, pages III:145–148, Piscataway, NJ, 1996. IEEE.
- Voss R. Random fractals: Characterization and measurement. In *Scaling Phenomena in Disordered Systems*. Plenum Press, New York, 1986.
- Vu N.-S., Dee H. M., and Caplier A. Face recognition using the POEM descriptor. *Pattern Recognition*, 45(7):2478–2488, 2012.
- Wang L. and He D. C. Texture classification using texture spectrum. *Pattern Recognition*, 23: 905–910, 1990.
- Wang S. and Rosenfeld A. A relative effectiveness of selected texture primitive. *IEEE Transactions on Systems, Man and Cybernetics*, 11:360–370, 1981.
- Wang Z., Guerriero A., and Sario M. D. Comparison of several approaches for the segmentation of texture images. *Pattern Recognition Letters*, 17:509–521, 1996.
- Weszka J. S., Dyer C., and Rosenfeld A. A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man and Cybernetics*, 6(4):269–285, 1976.
- Wood E. J. Applying Fourier and associated transforms to pattern characterization in textiles. *Textile Research Journal*, 60(4):212–220, 1990.
- Wu C. M., Chen Y. C., and Hsieh K. S. Texture features for classification of ultrasonic liver images. *IEEE Transactions on Medical Imaging*, 11:141–152, 1992.
- Xie X. A review of recent advances in surface defect detection using texture analysis techniques. *Electronic Letters on Computer Vision and Image Analysis*, 7:1–22, 2008.
- Xu Y., Sonka M., McLennan G., Guo J., and Hoffman E. MDCT-based 3-D texture classification of emphysema and early smoking related pathologies. *IEEE Transactions on Medical Imaging*, 25:464–475, 2006.
- Zhu Y. M. and Goutte R. A comparison of bilinear space/spatial-frequency representations for texture discrimination. *Pattern Recognition Letters*, 16:1057–1068, 1995.

Chapter 16

Motion analysis

Increasing processing power and, especially, the wide deployment of surveillance technology have made automated study of motion desirable and possible. Detection and tracking of human faces, pedestrian or vehicular motion are now common applications; additionally, object-based video compression, driver assistance, autonomous vehicles, robot navigation, user interfaces, smart room tracking, etc. can be seen. We have already considered the extraction of 3D shape and relative depth from motion in Section 12.1.1.

The usual input to a motion analysis system is a temporal image sequence. A set of assumptions can help to solve motion analysis problems—as always, prior knowledge helps to decrease the complexity of analysis. Prior knowledge may include information about the camera motion—mobile or static—and information about the time interval between consecutive images, especially whether this interval was short enough for the sequence to represent continuous motion. This prior information helps in the choice of an appropriate motion analysis technique. As in other areas of machine vision, there is no foolproof technique in motion analysis, no general algorithm; furthermore, the techniques presented in this chapter work only if certain conditions are met. A very interesting aspect of motion analysis is research into visual sensing of living organisms that are extremely well adapted to motion analysis. The psycho-physiological and cognitive aspects of motion sensing can be studied in [Ullman, 1979; Watson and Ahumada, 1985; Koenderink, 1986; Gescheider, 1997; Beutel et al., 2000; Cummins and Cummins, 2000; Kaernbach et al., 2003].

There are three main groups of motion-related problems from the practical point of view:

1. **Motion detection** is the simplest problem. This registers any detected motion and is often used for security purposes. This group usually uses a single static camera.
2. **Moving object detection and location** represents another set of problems. A camera is usually in a static location and objects are moving in the scene, or the camera moves and objects are static. These problems are considerably more difficult in comparison with the first group. If only moving object detection is required

(note the difference between motion detection and moving object detection), the solution can be based on motion-based segmentation methods. Other more complex problems include the detection of a moving object, the detection of the trajectory of its motion, and the prediction of its future trajectory. Image object-matching techniques are often used to solve this task—typically, direct matching in image data, matching of object features, matching of specific representative object points (corners, etc.) in an image sequence, or representing moving objects as graphs and consequent matching of these graphs. Another variety of approaches will parameterize the 2D projections of objects in view and make predictions about their movement, which are used and reinforced later in the image sequence. Practical examples of methods from this group include cloud tracking from a sequence of satellite meteorological data, including cloud character and motion prediction, motion analysis for autonomous road vehicles, automatic satellite location by detecting specific points of interest on the Earth's surface, city traffic analysis, and many military applications. The most complex methods of this group work even if both camera and objects are moving.

3. The third group is related to the **derivation of 3D object properties** from a set of 2D projections acquired at different instants. Three-dimensional object description is covered in Chapter 11 and an excellent survey of motion-based recognition approaches is given in [Cedras and Shah, 1995]. Practical solutions to 3D scene reconstruction from live video is described in [Akbarzadeh et al., 2006]

Even though motion analysis is often called *dynamic image analysis*, it is sometimes based on a small number of consecutive images, maybe just two or three in a sequence. This is similar to an analysis of static images, and the motion is actually analyzed at a higher level, looking for *correspondence* between pairs of points of interest in sequential images. A two-dimensional representation of a (generally) three-dimensional motion is called a *motion field*, in which each point is assigned a *velocity vector* corresponding to the motion direction, velocity, and distance from an observer at an appropriate image location.

A different approach analyzes motion from *optical flow* computation (Section 16.2), where a very small time distance between consecutive images is required, with very marginal change between two consecutive images. Optical flow computation results in motion direction and motion velocity determination at (possibly all) image points. The immediate aim of optical flow-based image analysis is to determine a motion field. As will be discussed later, optical flow does not always correspond to the true motion field because illumination changes are reflected in the optical flow. Object motion parameters can be derived from computed optical flow vectors. In reality, estimates of optical flow or point correspondence are noisy, but, unfortunately, three-dimensional interpretation of motion is ill-conditioned and requires high precision of optical flow or point correspondence. To overcome these difficulties, approaches that are not based on optical flow or point correspondence are used, since if the intermediate step (optical flow, point correspondence) does not have to be computed, possible errors can be avoided. Estimates of general motion of multiple moving objects in an image sequence based on gray-level and image gradient without using any higher-level information such as corners or borders is introduced in [Wu and Kittler, 1990]. Motion field construction using steerable filters also falls into this category [Freeman and Adelson, 1991; Huang and Chen, 1995]. A conceptually similar approach is presented in Section 16.4, in which image-based and

motion-based information are used simultaneously. By focusing on short-term motion patterns rather than on their tracking over extended periods of time, this approach does not require any complex intermediate representations. By analyzing temporal differences between shifted blocks of rectangle filters, it yields respectable performance in low quality small resolution images [Viola et al., 2003].

Motion field or *velocity field* computations represent a compromise technique; information similar to the optical flow is determined, but it is based on images acquired at intervals that are not short enough to ensure small changes due to motion. The velocity field can also be acquired if the number of images in a sequence is small.

Motion evaluation may or may not depend on object detection. An example of object-independent analysis is optical flow computation, whereas velocity field computation or differential methods search for points of interest or points of motion and represent object-dependent analysis. Object-dependent methods are usually based on searching for a correspondence between points of interest or between regions. One approach to motion analysis uses active contour models (snakes, discussed in Section 7.2). In motion analysis, the initial estimate necessary to start the snake energy minimization process is obtained from the detected position of the contour in the previous frame.

On the other hand, in situations when image-based information may be available to specifically characterize an object of interest, its usage is very helpful. Such information is used in kernel-based object detection and tracking as discussed in Section 16.5.2 [Comaniciu et al., 2000, 2003]. That method is based on spatially masking the target object with an isotropic kernel, followed by applying a smooth similarity function, which reduces the tracking problem to a maximum similarity search in the vicinity of the previous location. The similarity optimization is then efficiently performed using the mean shift algorithm (Section 7.1).

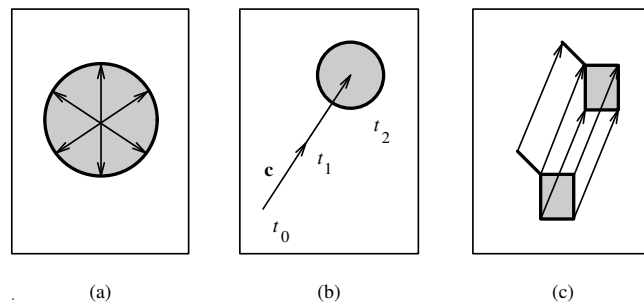


Figure 16.1: Object motion assumptions. (a) Maximum velocity (shaded circle represents area of possible object location). (b) Small acceleration (shaded circle represents area of possible object location at time t_2). (c) Common motion and mutual correspondence (rigid objects). © Cengage Learning 2015.

If motion analysis is based on detection of moving objects or object feature points, the following object motion assumptions can help to localize moving objects (Figure 16.1).

- **Maximum velocity:** Assume that a moving object is scanned at time intervals of dt . A possible position of a specific object point in an image is inside a circle with its center at the object point position in the previous frame and its radius $c_{\max} dt$, where $c_{\max}$ is the assumed maximum velocity of the moving object.

- **Small acceleration:** The change of velocity in time dt is bounded by some constant.
- **Common motion** (similarity in motion): All the object points move in a similar way.
- **Mutual correspondence:** Rigid objects exhibit stable pattern points. Each point of an object corresponds to exactly one point in the next image in sequence and vice versa, although there are exceptions due to occlusion and object rotation.

To generalize, image motion analysis and especially object tracking combine two separate but interrelated components:

- localization and representation of the object of interest (target), a mainly bottom-up process that needs to overcome inherent changes in the appearance, orientation, illumination, and scale of the target;
- trajectory filtering and data association, a top-down process that considers object dynamics and uses various sources of a priori information, as well as generation and evaluation of motion hypotheses, frequently using motion models.

Needless to say, stress may be on one or the other component based on the nature of the motion application. For example, tracking aircraft on radar or video images would depend heavily on the motion models associated with the individual aircraft types. In contrast, tracking human faces in a crowded scene where abrupt and unpredictable motion is more likely will probably depend more on target representation than on motion models. Equally important is to distinguish between single and multiple target tracking and design methods accordingly: all of these are addressed in the following sections. With motion analysis and object tracking being one of the most advanced application areas of machine vision, it shall be no surprise that methods, approaches, and strategies described in previous chapters will frequently be referenced and used as pre-requisite knowledge.

16.1 Differential motion analysis methods

Simple subtraction of images acquired at different instants in time makes motion detection possible, assuming a stationary camera position and constant illumination. A *difference image* $d(i, j)$ is a binary image where non-zero values represent image areas with motion, that is, areas where there was a substantial difference between gray-levels in consecutive images f_1 and f_2 :

$$\begin{aligned} d(i, j) &= 0 && \text{if } |f_1(i, j) - f_2(i, j)| \leq \varepsilon, \\ &= 1 && \text{otherwise,} \end{aligned} \tag{16.1}$$

where ε is a small positive number. Figure 16.2 shows an example of motion detection using a difference image. The difference image can be based on more complex image features such as average gray-level in some neighborhood, local texture features, etc. It is clear that the motion of any object distinct from its background can be detected (considering motion detection represents motion registration only).

Let f_1 and f_2 be two consecutive images separated by a time interval. An element $d(i, j)$ of the difference image between images f_1 and f_2 may have a value one due to any one of the following reasons (Figure 16.2):

1. $f_1(i, j)$ is a pixel on a moving object,
 $f_2(i, j)$ is a pixel on the static background,
(or vice versa).
2. $f_1(i, j)$ is a pixel on a moving object,
 $f_2(i, j)$ is a pixel on another moving object.
3. $f_1(i, j)$ is a pixel on a moving object,
 $f_2(i, j)$ is a pixel on a different part of the same moving object.
4. Noise, inaccuracies of stationary camera positioning, etc.

The system errors mentioned in the last item must be suppressed. The simplest solution is not to consider any regions of the difference image that are smaller than a specified threshold, although this may prevent slow motion and small object motions being detected. Further, results of this approach are highly dependent on an object–background contrast. On the other hand, we can be sure that all the resulting regions in the difference images result from motion. Section 16.5 considers these matters further.

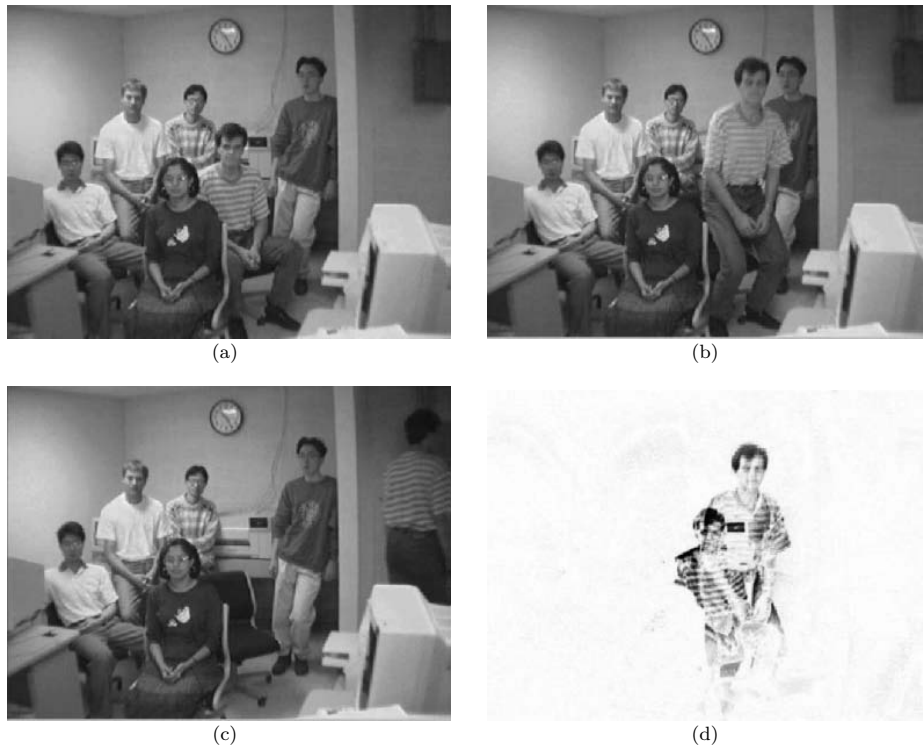


Figure 16.2: Motion detection. (a) First frame of the image sequence. (b) Frame 2 of the sequence. (c) Last frame (frame 5). (d) Differential motion image constructed from image frames 1 and 2 (inverted to improve visualization). © M. Sonka 2015.

Trajectories detected using differential image motion analysis may not reveal the direction of the motion. If direction is needed, construction of a *cumulative difference image* can solve this problem. Cumulative difference images contain information about motion direction and other time-related motion properties, and about slow motion and small object motion as well. The cumulative difference image d_{cum} is constructed from a sequence of n images, with the first image (f_1) being considered a reference image. Values of the cumulative difference image reflect how often (and by how much) the image gray-level was different from the gray-level of the reference image:

$$d_{\text{cum}}(i, j) = \sum_{k=1}^n a_k |f_1(i, j) - f_k(i, j)|, \quad (16.2)$$

a_k gives the *significance* of images in the sequence of n images; more recent images may be given greater weights to reflect the importance of current motion and to specify current object location. Figure 16.3 shows the cumulative difference image determined from the sequence of five image frames depicting motion analyzed in Figure 16.2.



Figure 16.3: Cumulative difference image determined from a sequence of five frames depicting motion analyzed in Figure 16.2 (inverted for improved visualization). © Cengage Learning 2015.

Suppose that an image of a static scene is available, and only stationary objects are present in it. If this image is used for reference, the difference image suppresses all motionless areas, and any motion in the scene can be detected as areas corresponding to the actual positions of the moving objects in the scene. Motion analysis can then be based on a sequence of difference images.

A problem with this approach may be the impossibility of getting an image of a static reference scene if the motion never ends; then a learning stage must construct the reference image. The most straightforward method is to superimpose moving image objects on non-moving image backgrounds from other images taken in a different phase of the motion. Which image parts should be superimposed can be judged from difference images, or the reference image can be constructed interactively (which can be allowed in the learning stage). Section 16.5.1 describes approaches to this problem.

Subsequent analysis usually determines motion trajectories; often only the centroid trajectory is needed. The task may be considerably simplified if objects can be segmented out of the first image of the sequence. A practical problem is the prediction of the motion trajectory if the object position in several previous images is known. There are many methods [Jain, 1981, 1984; Jain et al., 1995] that find other motion parameters from difference images—whether the object is approaching or receding, which object overlaps which, etc. Note that difference motion analysis methods give good examples of motion

analysis principles and present a good introduction to the problem; unfortunately, difference images do not carry enough information to work reliably in reality. Some problems are common for most motion field detection approaches—consider just a simple example of a rectangular object moving parallel to the object boundary; differential motion analysis can only detect the motion of two sides of the rectangle (see Figures 16.4a,b). Similarly, an aperture problem may cause ambiguity of contained motion information—in the situation shown in Figure 16.4c, only part of an object boundary is visible and it is impossible to determine the motion completely. The arrows indicate three possibilities of motion, all yielding the same final position of the object boundary in the image. Differential motion analysis is often used in digital subtraction angiography, where vessel motion is estimated.

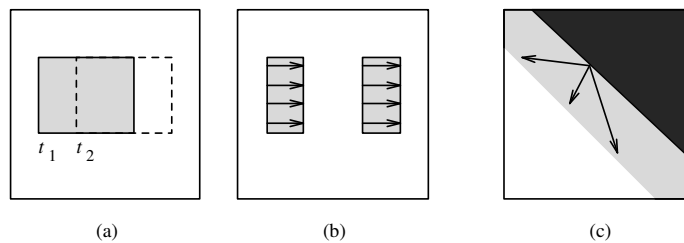


Figure 16.4: Problems of motion field construction. (a) Object position at times t_1 and t_2 . (b) Motion field. (c) Aperture problem—ambiguous motion. © Cengage Learning 2015.

While the difference image carries information about presence of motion, characteristics of motion derived from it are not very reliable. The motion parameter estimation robustness can be improved if intensity characteristics of regions or groups of pixels in two image frames are compared. A conceptually straightforward approach to *robust motion detection* is to compare corresponding areas of the image. Such corresponding *superpixels* are usually formed by non-overlapping rectangular regions, the size of which can be derived from the camera aspect ratio. Then, the superpixels may be matched in the compared frames using correlation or likelihood approaches [Jain et al., 1995].

Detecting *moving edges* helps overcome several limitations of differential motion analysis methods. By combining spatial and temporal image gradients, differential analysis can be used reliably for detection of slow-moving edges as well as of weak edges that move with higher speed. Moving edges can be determined by logical AND operations of the spatial and temporal image edges [Jain et al., 1979]. The spatial edges can be identified by virtually any edge detector from the variety given in Section 5.3.2, the temporal gradient can be approximated using the difference image, and the logical AND can be implemented through multiplication. Then, the moving edge image $d_{\text{med}}(i, j)$ can be determined as

$$d_{\text{med}}(i, j) = S(i, j) D(i, j), \quad (16.3)$$

where $S(i, j)$ represents edge magnitudes determined in one of the two analyzed image frames and $D(i, j)$ is the absolute difference image. An example of a moving-edge image determined from the first and second frames of an image sequence (Figure 16.2) is given in Figure 16.5.



Figure 16.5: Moving-edge image determined from the first and second frames of the image sequence analyzed in Figure 16.2 (inverted for improved visualization). © Cengage Learning 2015.

16.2 Optical flow

Optical flow reflects image changes due to motion during a time interval dt , and the optical flow field is the velocity field that represents the three-dimensional motion of object points across a two-dimensional image [Kearney and Thompson, 1988]. Optical flow is an abstraction typical of the kind that computational methods try to achieve. Therefore, it should represent only those motion-related intensity changes in the image that are required in further processing, and all other changes reflected in the optical flow should be considered detection errors. For example, optical flow should not be sensitive to illumination changes and motion of unimportant objects (e.g., shadows). However, non-zero optical flow is detected if a fixed sphere is illuminated by a moving source, and a smooth sphere rotating under constant illumination provides no optical flow despite the rotational motion and the true non-zero motion field [Horn, 1986]. Of course, the aim is to determine an optical flow that corresponds closely with the true motion field. Optical flow computation is a necessary precondition of subsequent higher-level processing that can solve motion-related problems if a camera is stationary or moving; it provides tools to determine parameters of motion, relative distances of objects in the image, etc. A simulated example of two consecutive images and a corresponding optical flow image are shown in Figure 16.6.

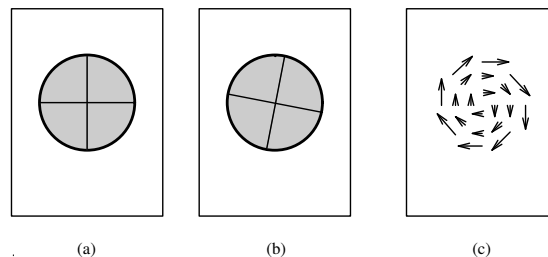


Figure 16.6: Optical flow. (a) Time t_1 . (b) Time t_2 . (c) Optical flow. © Cengage Learning 2015.

16.2.1 Optical flow computation

Optical flow computation is based on two assumptions:

1. The observed brightness of any object point is constant over time.

2. Nearby points in the image plane move in a similar manner (the *velocity smoothness* constraint).

Suppose we have a continuous image; $f(x, y, t)$ refers to the gray-level of (x, y) at time t . Representing a dynamic image as a function of position and time permits it to be expressed as a Taylor series:

$$f(x + dx, y + dy, t + dt) = f(x, y, t) + f_x dx + f_y dy + f_t dt + O(\partial^2), \quad (16.4)$$

where f_x, f_y, f_t denote the partial derivatives of f . We can assume that the immediate neighborhood of (x, y) is translated some small distance (dx, dy) during the interval dt ; that is, we can find dx, dy, dt such that

$$f(x + dx, y + dy, t + dt) = f(x, y, t). \quad (16.5)$$

If dx, dy, dt are very small, the higher-order terms in equation (16.4) vanish and

$$-f_t = f_x \frac{dx}{dt} + f_y \frac{dy}{dt}. \quad (16.6)$$

The goal is to compute the velocity

$$\mathbf{c} = \left(\frac{dx}{dt}, \frac{dy}{dt} \right) = (u, v), \quad (16.7)$$

f_x, f_y, f_t can be computed, or at least approximated, from $f(x, y, t)$. The motion velocity can then be estimated as

$$-f_t = f_x u + f_y v = \nabla f \cdot \mathbf{c}, \quad (16.8)$$

where ∇f is a two-dimensional image gradient. It can be seen from equation (16.8) that the gray-level difference f_t at the same location of the image at times t and $t + dt$ is a product of spatial gray-level difference and velocity in this location according to the observer.

Equation (16.8) does not specify the velocity vector completely; rather, it only provides the component in the direction of the brightest gradient (see Figure 16.4c). To solve the problem completely, a smoothness constraint is introduced; that is, the velocity vector field changes slowly in a given neighborhood. Full details of this approach may be found in [Horn and Schunk, 1981], but the approach reduces to minimizing the squared error quantity

$$E^2(x, y) = (f_x u + f_y v + f_t)^2 + \lambda(u_x^2 + u_y^2 + v_x^2 + v_y^2), \quad (16.9)$$

where $u_x^2, u_y^2, v_x^2, v_y^2$ denote partial derivatives squared as error terms. The first term represents a solution to equation (16.8), the second term is the smoothness criterion, and λ is a Lagrange multiplier. Using standard techniques [Horn and Schunk, 1981], this reduces to solving the differential equations

$$(\lambda^2 + f_x^2) u + f_x f_y v = \lambda^2 \bar{u} - f_x f_t, \quad (16.10)$$

$$f_x f_y u + (\lambda^2 + f_y^2) v = \lambda^2 \bar{v} - f_y f_t,$$

where $\bar{u}, \bar{v}$ are mean values of the velocity in directions x and y in some neighborhood of (x, y) . It can be shown that a solution to these equations is

$$u = \bar{u} - f_x \frac{P}{D}, \quad (16.11)$$

$$v = \bar{v} - f_y \frac{P}{D}, \quad (16.12)$$

where

$$P = f_x \bar{u} + f_y \bar{v} + f_t, \quad D = \lambda^2 + f_x^2 + f_y^2. \quad (16.13)$$

Determination of the optical flow is then based on a Gauss-Seidel iteration method using pairs of (consecutive) dynamic images [Horn, 1986; Young, 1971].

Algorithm 16.1: Relaxation computation of optical flow from dynamic image pairs

1. Initialize velocity vectors $\mathbf{c}(i, j) = 0$ for all (i, j) .
2. Let k denote the number of iterations. Compute values u^k, v^k for all pixels (i, j)

$$\begin{aligned} u^k(i, j) &= \bar{u}^{k-1}(i, j) - f_x(i, j) \frac{P(i, j)}{D(i, j)}, \\ v^k(i, j) &= \bar{v}^{k-1}(i, j) - f_y(i, j) \frac{P(i, j)}{D(i, j)}. \end{aligned} \quad (16.14)$$

Partial derivatives f_x, f_y, f_t can be estimated from pairs of consecutive images.

3. Stop if

$$\sum_i \sum_j E^2(i, j) < \varepsilon,$$

where ε is the maximum permitted error; return to step 2 otherwise.

If more than two images are to be processed, efficiency may be increased by using the results of one iteration to initialize the current image pair in sequence.

Algorithm 16.2: Optical flow computation from an image sequence

1. Evaluate starting values of the optical flow $\mathbf{c}(i, j)$ for all points (i, j) .
2. Let m be the sequence number of the currently processed image. For all pixels of the next image, evaluate

$$\begin{aligned} u^{m+1}(i, j) &= \bar{u}^m(i, j) - f_x(i, j) \frac{P(i, j)}{D(i, j)}, \\ v^{m+1}(i, j) &= \bar{v}^m(i, j) - f_y(i, j) \frac{P(i, j)}{D(i, j)}. \end{aligned} \quad (16.15)$$

3. Repeat step 2 to process all images in the sequence.

Both these algorithms are naturally parallel. The iterations may be very slow, with a computational complexity $\mathcal{O}(n^p)$, where p is the order of the partial differential equation set (16.10). Experimentally, it is found that thousands of iterations are needed until convergence if a second-order smoothness criterion is applied [Glazer, 1984]. On the other hand, the first 10–20 iterations usually leave an error smaller than the required accuracy, and the rest of the iterative process is then very gradual.

If the differences dx , dy , dt are very small, all the higher-order terms vanish in the continuous derivative of equation (16.4). Unfortunately, in reality this is often not the case if subsequent images are not taken frequently enough. As a result, the higher-order terms do not vanish and an estimation error results if they are neglected. To decrease this error, the second-order terms may be considered in the Taylor series, and the problem becomes a minimization of an integral over a local neighborhood N [Nagel, 1987]:

$$\iint_N \left[f(x, y, t) - f(x_0, y_0, t_0) - f_x[x - u] - f_y[y - v] - \frac{1}{2}f_{xx}[x - u]^2 - f_{xy}[x - u][y - v] - \frac{1}{2}f_{yy}[y - v]^2 \right]^2 dx dy. \quad (16.16)$$

This minimization is rather complex and may be simplified for image points that correspond to corners (Section 5.3.10). Let the co-ordinate system be aligned with the main curvature direction at (x_0, y_0) ; then $f_{xy} = 0$ and the only non-zero second-order derivatives are f_{xx} and f_{yy} . However, at least one of them must cross zero at (x_0, y_0) to get a maximum gradient: If, say, $f_{xx} = 0$, then $f_x \rightarrow \max$ and $f_y = 0$. With these assumptions, equation (16.16) simplifies, and the following formula is minimized [Vega-Riveros and Jabbour, 1989]:

$$\sum_{x, y \in N} \left[f(x, y, t) - f(x_0, y_0, t_0) - f_x(x - u) - \frac{1}{2}f_{yy}(y - v)^2 \right]^2. \quad (16.17)$$

A conventional minimization approach of differentiating equation (16.17) with respect to u and v and equating to zero results in two equations in the velocity components u, v .

16.2.2 Global and local optical flow estimation

Optical flow computation will be in error to the extent that the constant brightness and velocity smoothness assumptions are violated—unfortunately, in real imagery, this is quite common. Typically, the optical flow changes dramatically in highly textured regions, around moving boundaries, depth discontinuities, etc. [Kearney and Thompson, 1988]. A significant advantage of global relaxation methods of optical flow computation is to find the smoothest velocity field consistent with the image data; as discussed in Section 10.9, an important property of relaxation methods is their ability to propagate local constraints globally. As a result, not only constraint information but also all optical flow estimation errors propagate across the solution. Therefore, even a small number of problem areas in the optical flow field may cause widespread errors and poor estimates.

Since global error propagation is the biggest problem of the global optical flow computation scheme, local optical flow estimation may be a better solution. The local estimate is based on the same brightness and smoothness assumptions, and the idea is to divide the image into small regions where the assumptions hold. This solves the error propagation problem but another arises—in regions where spatial gradients change slowly, the

optical flow estimation becomes ill-conditioned because of lack of motion information, and it cannot be detected correctly. If a global method is applied to the same region, the information from neighboring image parts propagates and represents a basis for optical flow computation even if the local information was not sufficient by itself. The conclusion is that global sharing of information is beneficial in constraint sharing and detrimental with respect to error propagation [Kearney and Thompson, 1988].

One way to cope with the smoothness violation problem is to detect regions in which the smoothness constraints hold. Two heuristics for identifying neighboring constraint equations that differ substantially in their flow value were introduced in [Horn and Schunk, 1981]. The main problem is in selecting a threshold to decide which flow value

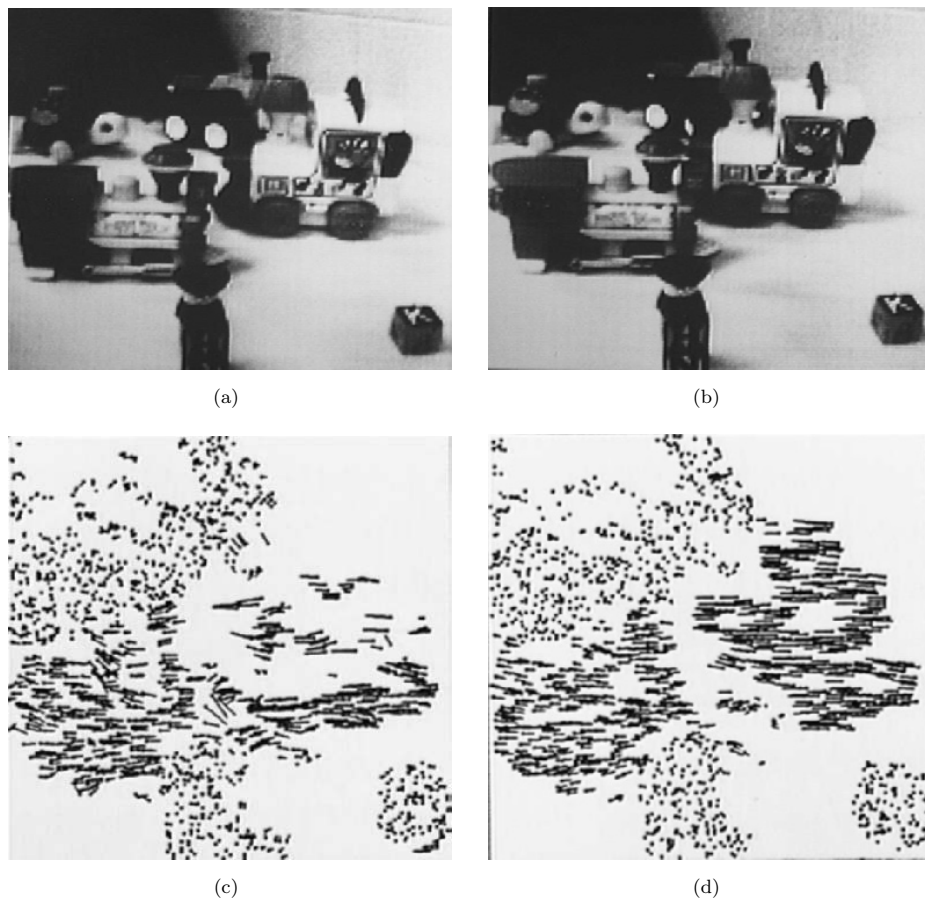


Figure 16.7: Moving trains image sequence. (a) First frame. (b) Last frame. (c) Optical flow detection—local optimization method. (d) Optical flow detection—method of continuous adaptation to errors. (Only 20% of vectors with moderate and high confidence shown.) *Courtesy of J. Kearney, The University of Iowa.*

difference should be considered substantial—if the threshold is set too low, many points are considered positioned along flow discontinuities, while if the threshold is too high, some points violating smoothness remain part of the computational net. The boundary between smooth subnets is not closed; paths between them remain, and the error propagation problem is not solved.

An approach of continuous adaptation to errors was introduced in [Kearney et al., 1987]. As with the basic global relaxation method, optical flow is determined iteratively by combining the local average flow vector with the gradient constraint equation. However, a confidence is assigned to each flow vector based on heuristic judgments of correctness, and the local average flow is computed as a weighted average by confidence. Thus, the propagation of error-free estimates is inhibited. Details of confidence estimation, smoothness violation detection, combining partial estimates, implementa-

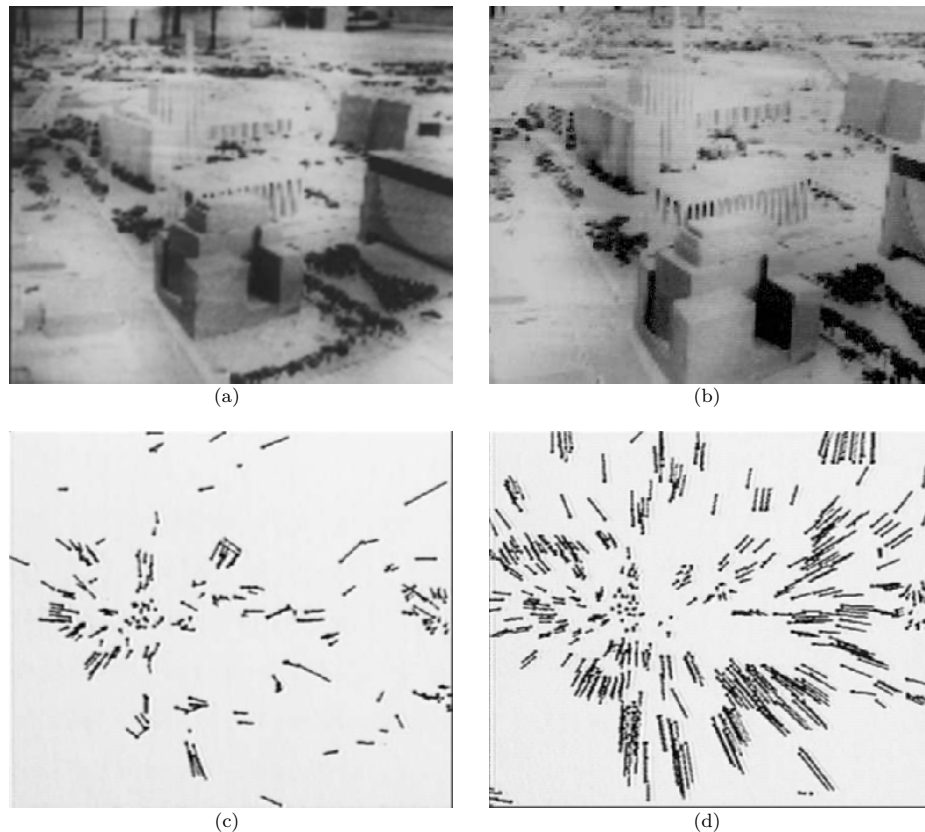


Figure 16.8: Simulated flyover image sequence. (a) First frame. (b) Last frame. (c) Optical flow detection—local optimization method. (d) Optical flow detection—method of continuous adaptation to errors. (Again, only 20% of vectors with moderate and high confidence shown.)
Courtesy of J. Kearney, The University of Iowa.

tion details, and discussion of results are given in [Kearney et al., 1987; Kearney and Thompson, 1988].

Performance of the method is illustrated in Figures 16.7 and 16.8. The first image pair, shown in Figures 16.7a,b, contains a collection of toys, and the second pair of images (Figures 16.8a,b) simulates a view from an aircraft flying over a city. Optical flow resulting from a simple local optimization is shown in Figures 16.7c and 16.8c, and results of the global method of continuous adaptation to errors are given in Figures 16.7d and 16.8d. The optical flow improvement achieved by the latter method is clearly visible.

Comparison of the performance of many optical flow techniques is given in [Barron et al., 1994]—local differential approaches [Fleet and Jepson, 1990; Lucas and Kanade, 1981] were found to be most accurate and robust. Techniques using global smoothness constraints were found to produce visually attractive motion fields, but give an accuracy suitable only for qualitative use, insufficient for ego-motion computation and 3D structure from motion detection.

16.2.3 Combined local–global optical flow estimation

Variational methods of optical flow computation, e.g., the classical Horn–Schunck approach described above using Gauss–Seidel iterative solution method is known to produce good results and dense flow fields, but suffer from slow convergence, rendering it unsuitable for real-time applications [Weickert and Schnoerr, 2001]. Combining the global Horn–Schunck variational approach with a local least-square Lucas–Kanade method [Lucas and Kanade, 1981], the approach named **combined local–global (CLG) method** was introduced in [Bruhn et al., 2002] which offers real-time performance [Bruhn et al., 2005]. The CLG method computes the optical flow field $(u, v)^T$ in a rectangular image domain Ω by minimization of the functional

$$E(u, v) = \int_{\Omega} (w^T J_{\rho}(\nabla_3 f) w + \alpha(|\nabla u|^2 + |\nabla v|^2)) \, dx dy, \quad (16.18)$$

where $w(x, y) = (u(x, y), v(x, y), 1)^T$ is the displacement, ∇u is the spatial gradient of $(u_x, u_y)^T$, $\nabla_3 f$ is the spatio-temporal gradient $(f_x, f_y, f_t)^T$, and α is a regularization parameter. $J_{\rho}(\nabla_3 f)$ is the structure tensor [Bigun et al., 1991] given by $K_{\rho} * (\nabla_3 f \nabla_3 f^T)$, where $*$ denotes convolution and K_{ρ} is a Gaussian with standard deviation ρ . Equation (16.18) represents a pure Horn–Schunck method for $\rho \rightarrow 0$ and the pure Lucas–Kanade method for $\alpha \rightarrow 0$. When the parameters are set to combine both approaches, good properties of both are maintained, resulting in dense flow fields and robustness to noise.

The energy functional minimization can be obtained by solving the Euler–Lagrange equations

$$\alpha \Delta u - (J_{11}(\nabla_3 f)u + J_{12}(\nabla_3 f)v + J_{13}(\nabla_3 f)) = 0, \quad (16.19)$$

$$\alpha \Delta v - (J_{21}(\nabla_3 f)u + J_{22}(\nabla_3 f)v + J_{23}(\nabla_3 f)) = 0, \quad (16.20)$$

where Δ denotes the Laplacian; boundary conditions shall reflect that normal derivative vanishes at boundaries of Ω

$$\partial_n u = 0, \partial_n v = 0. \quad (16.21)$$

Numerical solving of the above Euler–Lagrange equations can be performed, e.g., using the traditional Gauss–Seidel iteration approach [Young, 1971], but this approach is not

fast enough for real-time optical flow computation since after an initial stage of fast improvements by removing higher frequency error parts (via smoothing) it suffers from a slow process of achieving necessary accuracy since the available smoothing processes do not efficiently deal with the remaining low-frequency error components. Multigrid methods overcome the problem by using a coarse-to-fine hierarchy of equation systems delivering excellent error-reduction behavior [Briggs et al., 2000; Trottenberg et al., 2001]. Here, low frequencies on the finest grid reappear as higher frequencies on the coarser grids and can therefore be successfully removed. A two-orders of magnitude speed-up was reported by implementing the CLG method using a full multigrid method for solving the linear system of equations compared to commonly used variational techniques of optical flow computation [Bruhn et al., 2005].

16.2.4 Optical flow in motion analysis

Optical flow gives a description of motion and can be a valuable contribution to image interpretation even if no quantitative parameters are obtained from motion analysis. Optical flow can be used to study a large variety of motions—moving observer and static objects, static observer and moving objects, or both moving. Optical flow analysis does not result in motion trajectories as described in Section 16.1; instead, more general motion properties are detected that can significantly increase the reliability of complex dynamic image analysis [Thompson et al., 1985; Kearney et al., 1987; Aggarwal and Martin, 1988].

Motion, as it appears in dynamic images, is usually some combination of four basic elements:

- Translation at constant distance from the observer.
- Translation in depth relative to the observer.
- Rotation at constant distance about the view axis.
- Rotation of a planar object perpendicular to the view axis.

Optical-flow based motion analysis can recognize these basic elements by applying a few relatively simple operators to the flow. Motion form recognition is based on the following facts (Figure 16.9):

- Translation at constant distance is represented as a set of parallel motion vectors.
- Translation in depth forms a set of vectors having a common focus of expansion.
- Rotation at constant distance results in a set of concentric motion vectors.
- Rotation perpendicular to the view axis forms one or more sets of vectors starting from straight line segments.

Exact determination of rotation axes and translation trajectories can be computed, but with a significant increase in difficulty of analysis.

Consider translational motion: if the translation is not at constant depth, then optical flow vectors are not parallel, and their directions have a single **focus of expansion**

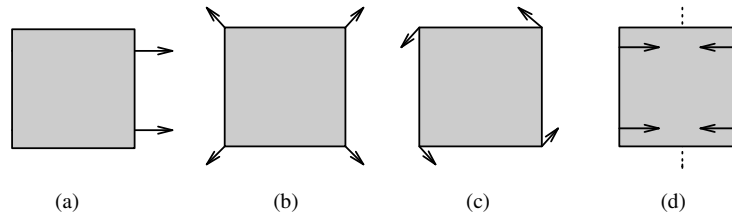


Figure 16.9: Motion form recognition. (a) Translation at constant distance. (b) Translation in depth. (c) Rotation at constant distance. (d) Planar object rotation perpendicular to the view axis. © Cengage Learning 2015.

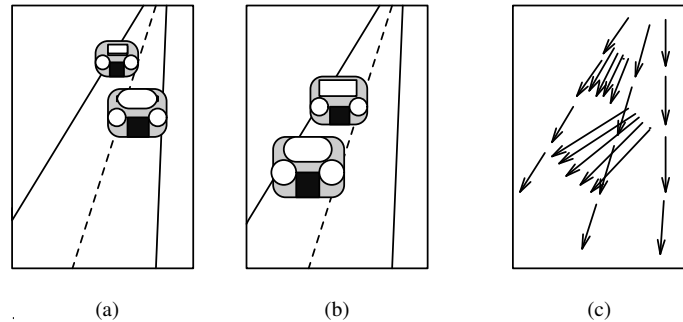


Figure 16.10: Focus of expansion. (a) Time t_1 . (b) Time t_2 . (c) Optical flow. © Cengage Learning 2015.

(FOE). If the translation is at constant depth, the FOE is at infinity. If several independently moving objects are present in the image, each motion has its own FOE—this is illustrated in Figure 16.10, where an observer moves in a car towards other approaching cars on the road.

Mutual velocity

The mutual velocity $\mathbf{c}$ of an observer and an object represented by an image point can be found in an optical flow representation. Let the mutual velocities in directions x, y, z be $c_x = u, c_y = v, c_z = w$, where z gives information about the depth (note that $z > 0$ for points in front of the image plane). To distinguish image co-ordinates in the following, let the image co-ordinates be x', y' . From perspective considerations, if (x_0, y_0, z_0) is the position of some point at time $t_0 = 0$, then the position of the same point at time t can, assuming unit focal distance of the optical system and constant velocity, be determined as follows:

$$(x', y') = \left(\frac{x_0 + ut}{z_0 + wt}, \frac{y_0 + vt}{z_0 + wt} \right). \quad (16.22)$$

FOE determination

The FOE in a two-dimensional image can be determined from this equation. Let us assume motion directed towards an observer; as $t \rightarrow -\infty$, the motion can be traced back

to the originating point at infinite distance from the observer. The motion towards an observer continues along straight lines and the originating point in the image plane is

$$\mathbf{x}'_{\text{FOE}} = \left(\frac{u}{w}, \frac{v}{w} \right). \quad (16.23)$$

Note that the same equation can be used for $t \rightarrow \infty$ and motion in the opposite direction. Clearly, any change of motion direction results in changes of velocities u, v, w , and the FOE changes its location in the image [Jain, 1983].

Distance (depth) determination

Because of the presence of a z co-ordinate in equation (16.22), optical flow can be used to determine the current distance of a moving object from the observer's position. The distance information is contained indirectly in equation (16.22). Assuming points of the same rigid object and translational motion, at least one actual distance value must be known to evaluate the distance exactly. Let $D(t)$ be the distance of a point from the FOE, measured in a two-dimensional image, and let $V(t)$ be its velocity dD/dt . The relationship between these quantities and the optical flow parameters is then

$$\frac{D(t)}{V(t)} = \frac{z(t)}{w(t)}. \quad (16.24)$$

This formula is a basis for determination of distances between moving objects. Assuming an object moving towards the observer, the ratio z/w specifies the time at which an object moving at a constant velocity w crosses the image plane. Based on the knowledge of the distance of any single point in an image which is moving with a velocity w along the z axis, it is possible to compute the distances of any other point in the image that is moving with the same velocity w

$$z_2(t) = \frac{z_1(t) V_1(t) D_2(t)}{D_1(t) V_2(t)}, \quad (16.25)$$

where $z_1(t)$ is the known distance and $z_2(t)$ is the unknown distance. Using the given formulae, relations between real-world co-ordinates x, y and image co-ordinates x', y' can be found related to the observer position and velocity:

$$x(t) = \frac{x'(t) w(t) D(t)}{V(t)}, \quad y(t) = \frac{y'(t) w(t) D(t)}{V(t)}, \quad z(t) = \frac{w(t) D(t)}{V(t)}. \quad (16.26)$$

Note that the above equations cover both the moving objects and the moving camera as long as the motion is along the camera optical axis. Situations in which motion is not realized along the optical axis are treated in [Jain et al., 1995].

Collision Prediction

A practical application is analysis of robot motion, where the optical flow approach is able to detect potential collisions with scene objects. Observer motion—as seen from optical flow representation—aims into the FOE of this motion; co-ordinates of this FOE are $(u/w, v/w)$. The origin of image co-ordinates (the imaging system focal point) proceeds

in the direction $\mathbf{s} = (u/w, v/w, 1)$ and follows a path in real-world co-ordinates at each time instant defined as a straight line,

$$(x, y, z) = t\mathbf{s} = t\left(\frac{u}{w}, \frac{v}{w}, 1\right), \quad (16.27)$$

where the parameter t represents time. The position of an observer $\mathbf{x}_{\text{obs}}$ when at its closest point of approach to some $\mathbf{x}$ in the real world is then

$$\mathbf{x}_{\text{obs}} = \frac{\mathbf{s}(\mathbf{s} \cdot \mathbf{x})}{\mathbf{s} \cdot \mathbf{s}}. \quad (16.28)$$

The smallest distance $d_{\min}$ between a point $\mathbf{x}$ and an observer during observer motion is

$$d_{\min} = \sqrt{(\mathbf{x} \cdot \mathbf{x}) - \frac{(\mathbf{x} \cdot \mathbf{s})^2}{\mathbf{s} \cdot \mathbf{s}}}. \quad (16.29)$$

Thus, a circular-shaped observer with radius r will collide with objects if their smallest distance of approach $d_{\min} < r$.

The analysis of motion, computation of FOE, depth, possible collisions, time to collision, etc., are all very practical problems. Interpretation of motion is discussed in [Subbarao, 1988], and motion analysis and computing range from an optical flow map is described in [Albus and Hong, 1990]. A comprehensive approach to motion parameter estimation from optical flow together with a comprehensive overview of existing techniques is given in [Hummel and Sundaeswaran, 1993]. Ego-motion estimation from optical flow fields determined from multiple cameras is presented in [Tsao et al., 1997]. Obstacle detection by evaluation of optical flow is presented in [Enkelmann, 1991]. Edge-based obstacle detection derived from determined size changes is presented in [Ringach and Baram, 1994]. Time to collision computation from first-order derivatives of image flow is described in [Subbarao, 1990], where it is shown that higher-order derivatives, which are unreliable and computationally expensive, are not necessary. Computation of FOE does not have to be based on optical flow; the spatial gradient approach and a natural constraint that an object must be in front of the camera to be imaged are used in a direct method of locating FOE in [Negahdaripour and Ganesan, 1992].

16.3 Analysis based on correspondence of interest points

Optical flow can be applied only if the intervals between image acquisitions are very short. If they are not, motion detection based on correspondence of **interest points (feature points)** is an alternative. Detection of corresponding object points in subsequent images is a fundamental part of this method—if this correspondence is known, velocity fields can easily be constructed (this does not consider the hard problem of constructing a dense velocity field from a sparse-correspondence-point velocity field).

The first step of the method is to find significant points in all images of the sequence—points least similar to their surrounding representing object corners, borders, or any other characteristic features in an image that can be tracked over time. Point detection is followed by a matching procedure, which looks for correspondences between these points. The process results in a sparse velocity field construction.

16.3.1 Detection of interest points

There is a host of interest point detectors that have developed over the years—see Section 5.3.10: an early one is the *Moravec*; well-used alternatives are the Kitchen-Rosenfeld and Zuniga-Haralick operators. The *Harris* corner detector (described in Section 5.3.10) has proved very popular.

16.3.2 Lucas-Kanade point tracking

Suppose we have two images I_1 and I_2 of a dynamic scene that are separated in time by a short interval, implying that individual points will not have moved far. If a pixel (x, y) in I_1 moves to $(x + u, y + v)$ in I_2 , we will be able to assume that u and v are ‘small’; we will also assume that the pixel remains unaltered in appearance (intensity), so

$$I_2(x + u, y + v) - I_1(x, y) = 0. \quad (16.30)$$

Since the displacement is small, we can make a linear approximation to $I_2(x + u, y + v)$ via a Taylor expansion:

$$I_2(x + u, y + v) \approx I_2(x, y) + \frac{\partial I_2}{\partial x}u + \frac{\partial I_2}{\partial y}v. \quad (16.31)$$

Combining equations (16.30) and (16.31) gives

$$(I_2(x, y) - I_1(x, y)) + \frac{\partial I_2}{\partial x}u + \frac{\partial I_2}{\partial y}v = 0, \quad (16.32)$$

an equation connecting the temporal and spatial differences around (x, y) , in which u and v are unknown, that is a restatement of equation (16.7).

Clearly we cannot solve for two unknowns from this single equation: but if we make the assumption that an immediate neighborhood of (x, y) behaves in the same manner, more equations can be derived. For example, if we assume that all of the 3×3 window centered at (x, y) moves (u, v) between frames, and does not change in intensity, then equation (16.32) provides nine linear equations for the two unknowns, and the system is now overdetermined. We can construct a 9×2 matrix A , each row of which is an estimate of $\frac{\partial I_2}{\partial x}$, $\frac{\partial I_2}{\partial y}$ at respective pixels, and a 9×1 vector $\mathbf{b}$, each component of which is the difference in intensities between I_1, I_2 at respective pixels:

$$A \begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{b}.$$

A least-squares ‘best’ solution to this system is then available from

$$A^T A \begin{pmatrix} u \\ v \end{pmatrix} = A^T \mathbf{b}, \quad (16.33)$$

which is a 2×2 system, solvable if $A^T A$ is invertible.

Writing for brevity $I_x = \frac{\partial I_2}{\partial x}$ (and similar), we can see that

$$A^T A = \begin{pmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{pmatrix},$$

where the sum is taken over the small local neighborhood—the Harris matrix introduced as equation (5.73). As in the discussion there, this approach requires stability in $(A^T A)^{-1}$, which is provided by the two eigenvalues being ‘not too small’, and of comparable magnitude to each other. This will imply some structure (a corner, or similar feature) at the pixel in question, thereby avoiding the aperture problem described in Section 5.3.10. In practice, we might compute the minimum eigenvalue λ_m of $(A^T A)^{-1}$ at each pixel, select via some threshold those which are ‘large enough’, then retain pixels at which λ_m is locally maximal, and then retain all such pixels that are separated by some minimum distance.

The reasons that this approach might fail are a violation of its assumptions: maintaining local intensity, or significant local motion. The latter problem can often be addressed by an iterative approach: velocity estimates are made at pixels in I_1 , which is then warped towards I_2 , and the procedure is then repeated.

This approach to point tracking was first outlined in 1981 [Lucas and Kanade, 1981], and was developed with widespread impact into the *KLT tracker* [Shi and Tomasi, 1994] (which is freely available in many implementations on the WWW.) This work successfully uses the Lucas–Kanade approach to determine automatically in an image sequence points which are ‘trackable’ (so satisfy the criteria outlined above), and which have ‘real’ meaning in the scene. The key to this is recognizing that simple translation is not likely to be a good representation of motion in even small spatial windows, and that an affine model is likely to be much better—in the notation used above, we conjecture a 2×2 matrix D such that (x, y) in I_1 moves to

$$D \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} u \\ v \end{pmatrix}$$

in I_2 . The pure translational model is still used for tracking frame-to-frame, but the affine model can be used to confirm that the feature being tracked is ‘good’. D can be recaptured by a least-squares approach very similar to that used to solve for pure translation; then the dissimilarity (sum of square differences) between the original window and its [affine] transformed version can be considered. This successfully handles change of scale of features as camera distance changes, but will detect, for example, that a strong feature that is an accident of viewpoint (two strong boundaries crossing, but belonging to different scene features) is not a good candidate for tracking as viewpoint evolves.

Since its first publication, Lucas–Kanade flow computation, and associated trackers, have acquired huge popularity. A good survey of the algorithm’s developments and generalizations may be found in [Baker and Matthews, 2004]. In its most general form to incrementally warp—not necessarily in an affine way—an image I to a template T , the algorithm may be summarized as:

Algorithm 16.3: General Lucas–Kanade tracking

1. Initialize a warp $\mathbf{W}(\mathbf{x}; \mathbf{p})$, determined by some parameters $\mathbf{p}$.
2. Warp I with $\mathbf{W}(\mathbf{x}; \mathbf{p})$ to $\hat{I}$, and determine the error $T - \hat{I}$.

3. Warp the gradient of I , ∇I with $\mathbf{W}(\mathbf{x}; \mathbf{p})$, and evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$. Compute the ‘steepest descent’ image $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$, and the *Hessian* matrix

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right].$$

4. Compute

$$\Delta \mathbf{p} = H^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T [T(\mathbf{x}) - \hat{I}].$$

5. Set

$$\mathbf{p} = \mathbf{p} + \Delta \mathbf{p}$$

and go to 2 until $\|\Delta \mathbf{p}\| < \epsilon$.

This is, as in the more elementary exposition given above, a least-squares minimization of an error measure, applied iteratively until convergence. The only constraint put on the warping function $\mathbf{W}$ is that it be differentiable with respect to its parameters $\mathbf{p}$. Baker and Matthews give a comprehensive discussion of convergence, efficiency and other issues [Baker and Matthews, 2004].¹

An example of KLT tracking is given in Figure 16.11, where motion of ‘good’ points over 0.5 seconds is illustrated: these data were subsequently used as input to a system that built semantic models of scenes (here a Metro station) from unconstrained video [Dee et al., 2012].



Figure 16.11: KLT tracking applied to a video scene of a Metro station: the algorithm automatically determines trackable points that persist over the sequence. These ‘tracklets’ can be used subsequently to derive a semantic description of the scene. *Courtesy of H. M. Dee, The University of Aberystwyth. A color version of this figure may be seen in the color inset—Plate 36.*

16.3.3 Correspondence of interest points

Assuming that interest points have been located in all images of a sequence, a correspondence between points in consecutive images is sought. Many approaches may be applied

¹There is an associated website http://www.ri.cmu.edu/projects/project_515.html providing a variety of Matlab source codes and test images.

to seek an optimal correspondence, and several possible solutions have been presented earlier (Chapters 9 and 11). The graph matching problem, stereo matching, and ‘shape from X’ problems treat essentially the same problem.

An early, probabilistic method [Thompson and Barnard, 1981] is a good example of the main ideas of this approach: the correspondence search process is iterative and begins with the detection of all potential correspondence pairs in consecutive images. A maximum velocity assumption can be used for potential correspondence detection, which decreases the number of possible correspondences, especially in large images. Each pair of corresponding points is assigned a number representing the probability of correspondence. These probabilities are then iteratively recomputed to get a globally optimum set of pairwise correspondences (the maximum probability of pairs in the whole image, equation 16.39) using another assumption: **common motion principle**. The process ends if each point of interest in a previous image corresponds with precisely one point of interest in the next **and**

- The global probability of correspondences between image point pairs is significantly higher than other potential correspondences.
- Or the global probability of correspondences of points is higher than a pre-selected threshold.
- Or the global probability of correspondences gives a maximum probability (optimum) of all possible correspondences (note that $n!$ possible correspondences exist for n pairs of interest points).

Let $A_1 = \{\mathbf{x}_m\}$ be the set of all interest points in the first image, and $A_2 = \{\mathbf{y}_n\}$ the interest points of the second image. Let $\mathbf{c}_{mn}$ be a vector connecting points $\mathbf{x}_m$ and $\mathbf{y}_n$ ($\mathbf{c}_{mn}$ is thus a velocity vector; $\mathbf{y}_n = \mathbf{x}_m + \mathbf{c}_{mn}$). Let the probability of correspondence of two points $\mathbf{x}_m$ and $\mathbf{y}_n$ be P_{mn} . Two points $\mathbf{x}_m$ and $\mathbf{y}_n$ can be considered potentially corresponding if their distance satisfies the assumption of maximum velocity

$$|\mathbf{x}_m - \mathbf{y}_n| \leq c_{\max}, \quad (16.34)$$

where $c_{\max}$ is the maximum distance a point may move in the time interval between two consecutive images. Two correspondences of points $\mathbf{x}_m \mathbf{y}_n$ and $\mathbf{x}_k \mathbf{y}_l$ are termed consistent if

$$|\mathbf{c}_{mn} - \mathbf{c}_{kl}| \leq c_{\text{dif}}, \quad (16.35)$$

where c_{dif} is a preset constant derived from prior knowledge. Clearly, consistency of corresponding point pairs increases the probability that a correspondence pair is correct. This principle is applied in Algorithm 16.4 [Barnard and Thompson, 1980].

Algorithm 16.4: Velocity field computation from two consecutive images

1. Determine the interest points A_1 and A_2 in images f_1, f_2 , and detect all potential correspondences between point pairs $\mathbf{x}_m \in A_1$ and $\mathbf{y}_n \in A_2$.
2. Construct a data structure in which potential correspondence information of all points $\mathbf{x}_m \in A_1$ with points $\mathbf{y}_n \in A_2$ is stored, as follows

$$[\mathbf{x}_m, (\mathbf{c}_{m1}, P_{m1}), (\mathbf{c}_{m2}, P_{m2}), \dots, (V^*, P^*)]. \quad (16.36)$$

P_{mn} is the probability of correspondence of points $\mathbf{x}_m$ and $\mathbf{y}_n$, and V^* and P^* are special symbols indicating that no potential correspondence was found.

3. Initialize the probabilities P_{mn}^0 of correspondence based on local similarity—if two points correspond, their neighborhood should correspond as well:

$$P_{mn}^0 = \frac{1}{(1 + kw_{mn})}, \quad (16.37)$$

where k is a constant and

$$w_{mn} = \sum_{\Delta \mathbf{x}} [f_1(\mathbf{x}_m + \Delta \mathbf{x}) - f_2(\mathbf{y}_n + \Delta \mathbf{x})]^2, \quad (16.38)$$

$\Delta \mathbf{x}$ defines a (usually symmetric) neighborhood for match testing by ranging between some small negative and positive values.

4. Iteratively determine the probability of correspondence of a point $\mathbf{x}_m$ with points $\mathbf{y}_n$ as a weighted sum of probabilities of correspondence of all *consistent* pairs $\mathbf{x}_k \mathbf{y}_l$, where $\mathbf{x}_k$ are neighbors of $\mathbf{x}_m$ and the consistency of $\mathbf{x}_k \mathbf{y}_l$ is evaluated according to $\mathbf{x}_m, \mathbf{y}_n$. The *quality* q_{mn} of the correspondence pair is

$$q_{mn}^{(s-1)} = \sum_k \sum_l P_{kl}^{(s-1)}, \quad (16.39)$$

where s denotes an iteration step, k refers to all points $\mathbf{x}_k$ that are neighbors of $\mathbf{x}_m$, and l refers to all points $\mathbf{y}_l \in A_2$ that form pairs $\mathbf{x}_k \mathbf{y}_l$ consistent with $\mathbf{x}_m \mathbf{y}_n$.

5. Update the probabilities of correspondence for each point pair $\mathbf{x}_m, \mathbf{y}_n$

$$\hat{P}_{mn}^{(s)} = P_{mn}^{(s-1)} \left(a + bq_{mn}^{(s-1)} \right), \quad (16.40)$$

where a and b are preset constants. Normalize

$$P_{mn}^s = \frac{\hat{P}_{mn}^s}{\sum_j \hat{P}_{mj}^s}. \quad (16.41)$$

6. Repeat steps 4 and 5 until the best correspondence $\mathbf{x}_m \mathbf{y}_n$ is found for all points $\mathbf{x}_m$.
7. Vectors $\mathbf{c}_{ij}$ of the correspondence form a velocity field of the analyzed motion.

The velocity field resulting from this algorithm applied to the image pairs given in Figures 16.7a,b and 16.8a,b are shown in Figure 16.12. Note that the results are much better for the train sequence; compare the flyover velocity field with the optical flow results given in Figure 16.8d.

Velocity fields can be applied in position prediction tasks as well as optical flow. A good example of interpretation of motion derived from detecting interest points is

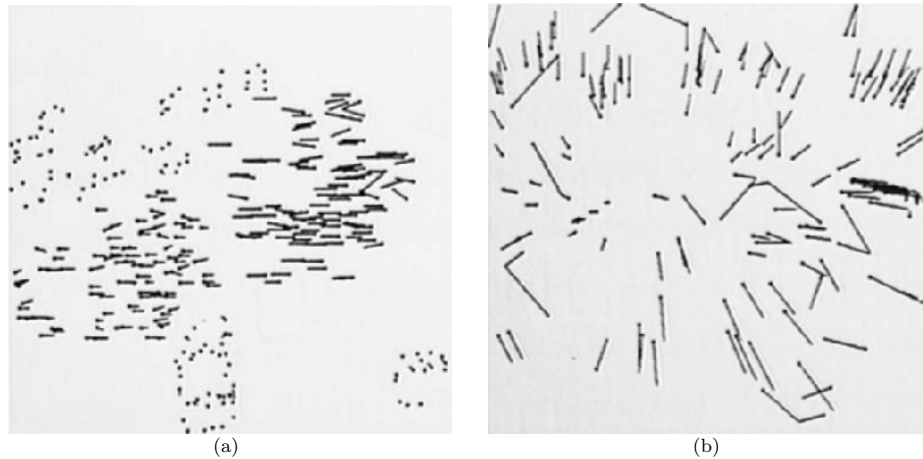


Figure 16.12: Velocity fields of the train sequence (left) and flyover (right) (original images shown in Figures 16.7a,b and 16.8a,b). *Courtesy of J. Kearney, The University of Iowa.*

given in [Scott, 1988]. Detection of moving objects from a moving camera using point correspondence in two orthographic views is discussed in [Thompson et al., 1993]. Fluid motion analysis using particle correspondence and dynamic programming is described in [Shapiro et al., 1995]. Two algorithms applicable to motion analysis in long monocular image sequences were introduced in [Hu and Ahuja, 1993].

Approaches that allow object registration without determination of explicit point correspondences exist. In [Fua and Leclerc, 1994], a method using full three-dimensional surface models is presented that may be used together with shape from motion analysis. An accurate and fast method for motion analysis that seeks correspondence of moving objects via a multi-resolution Hough transform is introduced in [Bober and Kittler, 1994].

16.4 Detection of specific motion patterns

In many cases, we are interested in detecting a specific class of motion, in which case some motion-specific information may be derived from a training set of examples and a classifier trained to distinguish between this and other phenomena that can be observed in the image sequences. The approach described below is motivated by detection of pedestrian motion but it can be applied to a variety of other applications.

Pedestrian motion detection and tracking are important tasks in surveillance applications. In non-pedestrian motion-related applications, training a target detector on a set of examples often yields an efficient detector with success shown for detecting cars, human faces, etc. Such detectors scan through an image looking for a match between the detector and the input image data. The candidate objects can then be tracked over time, further increasing the reliability of detection and the associated tracking. Several approaches are discussed below in Section 16.5. Many such methods first analyze the image-based information that is later processed using motion analysis techniques. These approaches require complex intermediate representations and perform matching,

segmentation, alignment, registration, and motion analysis. Since the detection/tracking is carried out in an open loop, failures occurring in the earlier steps may affect the performance in the later stages.

While these approaches see useful performance in some applications, they are not very suitable for pedestrian motion detection and tracking due to the inherent variability of body pose and clothing. The task is made more difficult by the typically low resolution character of the analyzed images: surveillance cameras may depict pedestrians in image patches as small as 20×10 pixels, yet human motion patterns are easily distinguishable even in such low-resolution data. One such system assesses motion periodicity directly from the tracked images and analyzes long temporal sequences to reject candidates with inconsistent motion patterns [Cutler and Davis, 2000].

A substantially different approach was presented in [Viola et al., 2003], in which image-based and motion-based information are used simultaneously. This method focuses on detection of short-term motion patterns rather than on tracking over extended periods of time. This work is closely related to the object detection method described in Section 10.7 and uses the AdaBoost learning paradigm (Section 9.8). As in Section 10.7, pedestrian motion detection uses a small set of simple rectangle filters trained on a set of examples, and can be evaluated very efficiently at any scale (Section 4.2.1). To capture the motion pattern, the filters need to work with short temporal image sequences.

In general, motion can be detected by determination of temporal differences in corresponding image blocks. The size of such image blocks determines the analysis scale, and for multi-scale analysis, blocks of different sizes are considered. Clearly, optical flow (Section 16.2) is able to provide the necessary information. However, the relatively high computational demands of optical flow calculation is not well suited for multi-scale analysis. On the other hand, the computational efficiency of rectangular filters and their multi-scale capabilities are well suited to this task. Simply considering differences between scale-specific blocks in pairs of images over time can serve as an identifier of motion (Section 16.1). Additionally, motion direction can be derived from differences between shifted image blocks, one from the image frame acquired at time t , the other at time $t + \delta t$. Five such images were shown as highly relevant

$$\begin{aligned}\Delta &= \text{abs}(I_t - I_{t+1}) , \\ U &= \text{abs}(I_t - I_{t+\delta t} \uparrow) , \\ D &= \text{abs}(I_t - I_{t+\delta t} \downarrow) , \\ L &= \text{abs}(I_t - I_{t+\delta t} \leftarrow) , \\ R &= \text{abs}(I_t - I_{t+\delta t} \rightarrow) ,\end{aligned}\tag{16.42}$$

where the arrows represent the direction of the image shift. For example, $I_{t+1} \downarrow$ refers to an image frame $t + \delta t$ shifted down by ψ pixels with respect to the previous image frame I_t . Figure 16.13 shows examples of the Δ, U, D, L, R images.

Similarly, filters f_k measure the magnitude of motion.

$$f_k = r_k(S) .\tag{16.43}$$

Several filter types can be designed. Filters

$$f_i = r_i(\Delta) - r_i(S)\tag{16.44}$$

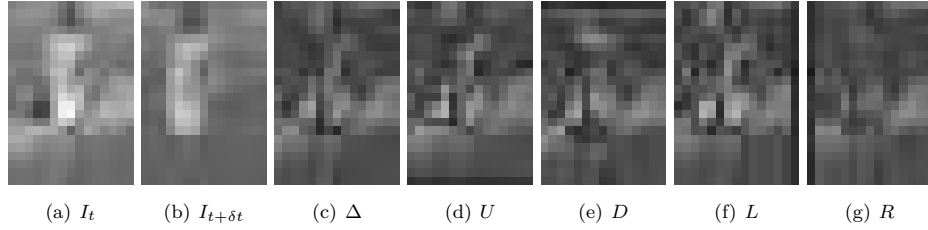


Figure 16.13: Motion and appearance difference images derived according to equation (16.42). Image R has the lowest energy and as such, corresponds to the right-to-left direction of motion. © M. Sonka 2015.

reflect the likelihood that a particular region is moving in a tested direction $\uparrow, \downarrow, \leftarrow, \rightarrow$. Here, S is one of the difference images $\{U, D, L, R\}$ and r_i is a single rectangle sum within the detection window.

Motion shear can be determined using filters

$$f_j = \phi_j(S) . \quad (16.45)$$

Rectangle appearance filters f_m contribute to detecting image patterns of expected static image properties

$$f_m = \phi(I_t) \quad (16.46)$$

from the first frame of the two-frame image sequence. As before, the filters $f_\bullet$ can be efficiently evaluated using the integral image representation described in Section 4.2.1.

Similarly to preceding work [Viola and Jones, 2001], the filters $f_\bullet$ can be of any size, aspect ratio, or position as long as they fit in the detection image block. As such, a large number of filters can be designed and a best subset of filters and a corresponding classifier must be designed to separate the moving objects with motion-specific properties from the rest of the image. The classifier C consists of a linear combination of the selected features and, after the AdaBoost training phase, reflects a thresholded sum of features

$$\begin{aligned} C(I_t, I_{t+\delta t}) &= 1 \quad \text{if} \quad \sum_{s=1}^N F_s(I_t, I_{t+\delta t}) > \theta , \\ &= 0 \quad \text{otherwise.} \end{aligned} \quad (16.47)$$

The feature F_s is a thresholded image that outputs one of two possible values

$$\begin{aligned} F_s(I_t, I_{t+\delta t}) &= \alpha \quad \text{if} \quad f_s(I_t, I_{t+\delta t}, \Delta, U, D, L, R) > t_s , \\ &= \beta \quad \text{otherwise,} \end{aligned} \quad (16.48)$$

where $t_s \in \mathcal{R}$ is a feature threshold and f_s is one of the filters $f_\bullet$. As before, the N features f_s are selected using AdaBoost from all possible filters, these being a function of one or more parameters $I_t, I_{t+\delta t}, \Delta, U, D, L$, and/or R , respectively. The values of α, β, t_s , and θ are computed during the AdaBoost training process (Section 9.8). Each of the N rounds of AdaBoost chooses from the full set of motion and appearance features. As a result, a mix of features balancing the appearance and motion descriptors is selected.

To support multi-scale detection, the parameter ψ used for calculating the shifted images in directions $\uparrow, \downarrow, \leftarrow, \rightarrow$ must be defined with respect to the detection scale. Consequently, motion-invariant detection of object motion speed is achieved. This invariance can be obtained in the training process by scaling all training samples to a pre-determined base resolution (i.e., bounding block size with respect to pixel counts in x and y directions). In [Viola et al., 2003], a base resolution of 20×15 pixels was used. During analysis, multi-scale behavior is achieved by operating on image pyramids (Section 4.3.1), which are computed from the pyramidal representations of the analyzed image frames I_t^l and $I_{t+\delta t}^l$

$$\begin{aligned}\Delta^l &= \text{abs}(I_t^l - I_{t+1}^l), \\ U^l &= \text{abs}(I_t^l - I_{t+\delta t}^l \uparrow), \\ D^l &= \text{abs}(I_t^l - I_{t+\delta t}^l \downarrow), \\ L^l &= \text{abs}(I_t^l - I_{t+\delta t}^l \leftarrow), \\ R^l &= \text{abs}(I_t^l - I_{t+\delta t}^l \rightarrow),\end{aligned}\tag{16.49}$$

where l denotes the pyramid level. Features are computed from the pyramidal representations in a scale-invariant fashion. In [Viola et al., 2003], a scale factor of 0.8 was used to generate successive pyramid levels all the way down to the pre-determined size of the base-resolution image block (20×15 pixels in the discussed case).

Once the features are selected, a boosted cascade of classifiers (Section 10.7, Algorithm 10.13 is used to increase the detection efficiency (see Figure 10.34). Simple classifiers with high detection rates and relatively high false positive rates are employed in the early stages of the boosted cascade, with more complex classifiers using larger numbers of features being used in later stages. Each stage of the cascade attempts to reduce both the detection and the false positive rates, obviously with a goal of reducing the false positive rate more rapidly than the detection rate.

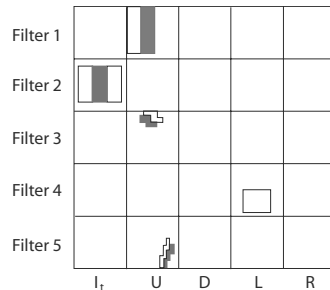


Figure 16.14: The first 5 features identified by the feature selection process for detecting walking pedestrians. Features reflect that pedestrians were centered in the training images, tend to be different from background, and four of them use the motion-difference images. Based on [Viola et al., 2003].

In the example application, a dynamic pedestrian detector was learned from sequences consisting of 2,000 frames. Each of the cascade classifiers was trained on 2,250 positive and 2,250 negative examples, each example consisting of two 20×15 image windows from two consecutive image frames ($\delta t = 1$). The positive examples were selected as scaled bounding boxes of the pedestrians, the negative examples did not include pedestrians. During the feature selection, 54,624 filters were considered, uniformly sub-sampled from a much larger set of all possible filters that can be defined on the 20×15 subwindow. Figure 16.14 shows the first 5 features learned. As further demonstrated

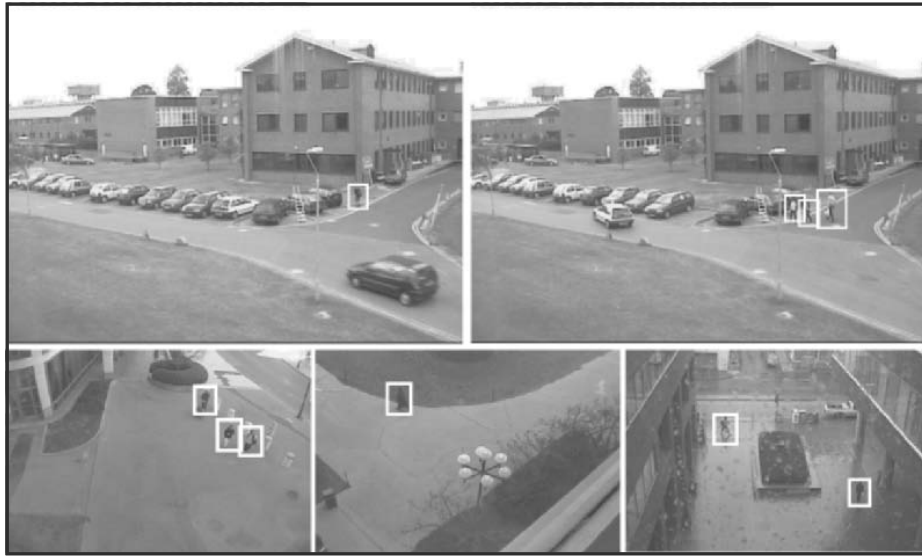


Figure 16.15: Results of pedestrian detection using dynamic pedestrian detector. *Courtesy of P. Viola, Microsoft Live Labs and M. Jones, Mitsubishi Electric Research Labs. © 2003 IEEE. Reprinted, with permission, from P. Viola, M. J. Jones, D. Snow, “Detecting Pedestrians Using Patterns of Motion and Appearance,” Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003), IEEE, 2003.*

in [Viola et al., 2003], the motion information was crucial for the achieved performance of the pedestrian detector, the dynamic pedestrian detector clearly outperformed the static pedestrian detector built using the generally identical boosted cascade of classifiers approach, but not using the motion-difference image information. Figure 16.15 shows several typical detection outcomes of the presented method.

16.5 Video tracking

Surveillance CCTV is becoming or has become ubiquitous in many countries. Recognizing that only a tiny percentage of this video is viewed ‘live’, this has generated enormous interest in computer vision algorithms that can monitor scenes automatically. While the application value (albeit sinister) of such work in monitoring the movements of people and traffic is obvious, there is a good number of related applications that might use the same algorithms and technology. Popular examples are in the agricultural domain for animal tracking, and sport, where successful real-time tracking is very lucrative, providing the potential for developments in training techniques: see, for example, [Intille and Bobick, 1995; Magee and Boyle, 2002; Needham and Boyle, 2001].

In fact, this area is very wide and provokes many deep and interesting problems both within and without vision. At simplest, we wish to take video signals, preferably in real-time, and determine activity of interest within them (‘interest’ here is obviously domain specific)—these are areas where computer vision can assist. The problem may become as complex as we wish by considering pan-tilt-zoom (PTZ) cameras, and cameras with

overlapping fields of view. Thereafter, we may be interested in modeling and observing behavior (another very subjective word)—in municipal scenes we might wish to identify criminal behavior; in traffic scenes we might wish to identify congestion or accidents; in sports scenes we might wish to identify particular strategies.

Unsurprisingly, these issues are key in robotics where agents may monitor their progress and surroundings visually. This is a burgeoning research area that this book does not address, but key within it is the concept of **Simultaneous Localization and Mapping**, or **SLAM**. SLAM recognizes that robots may use ‘maps’ of their environment to aid navigation and obstacle avoidance, but that pre-compiled maps may well be inaccurate, or evolve, or indeed may not exist. Accordingly, the robot constructs a map as it moves. It will be clear that this is a problem of some depth, dependent on an understanding of the agent’s kinematics, its models of what it may ‘see’ and its ability to re-recognize scenes from new viewpoints. It is often presented as a ‘chicken and egg’ issue—the robot cannot move securely without the map, but the map cannot be built until the robot has moved. In all probability, the robot will use a variety of sensors (in addition to visual) and so a full coverage is beyond this text—nevertheless, SLAM-based systems will use a number of the topics of this chapter as components. Just one example is robotic exploration of the sea-bed wreck of the Titanic, where multiple visual images often have small overlap, but the surveying vehicle’s inertial sensors provide independent input to mapping [Eustice et al., 2005]. The interested reader is directed toward robotics literature, and any number of useful WWW-based tutorials.

In this section we shall outline how a video signal might be processed to deliver data on blobs (objects) moving in the scene. Conceptually a simple task, it proves very challenging in reality and has benefited enormously from theoretical work coupled with increasing processor power.

16.5.1 Background modeling

The obvious approach to monitoring video is to ask what is moving in the scene, which is most simply done by comparing what we see with an empty, background scene, and performing some sort of comparison or subtraction. A video represents a sequence of frames (images)—much the simplest approach to modeling background is to consider the first frame to *be* the background, and to subtract intensities pixel by pixel in all successors. Non-zero differences then represent movement; these may be thresholded (with varying degrees of sophistication) and grouped (perhaps using morphological approaches) to derive blobs that will be the objects moving in the scene (Section 16.1).

It is easy to see why this will not normally work. Such an approach detects every single movement, howsoever small—this would include wind shaking trees, and the tiniest camera movement. More subtly, changes in lighting of any degree would have catastrophic effects. We would only take this very simple approach in conditions under very strict control; an absolutely rigid camera with no environmental variation in lighting or shadow, and minimal or absent moving ‘clutter’. These constraints obviously exclude most scenes of any real interest.

More reasonably, we might concede that ‘background’ is not constant and try to maintain a background frame dynamically. Success has been had with this approach by taking the mean or (more usually) median intensity at each pixel of the last K frames. This will have the effect, over a period K , of allowing a per-pixel update of lighting

changes, although of course a slow moving object of uniform intensity might then become absorbed into background.

Algorithm 16.5: Background maintenance by median filtering

1. Initialize: Acquire K frames. At each pixel, determine the median intensity of these K . If the image sequence is in color, this is done for each of the R,G,B streams. The median represents the current background value.
2. Acquire frame $K + 1$. Compute the difference between this frame and the current background at each pixel (a scalar for a gray image; a vector for RGB).
3. Threshold this difference to remove/reduce noise. Simple thresholds may be too crude at this point and there is scope for using, e.g., hysteresis (Algorithm 6.5).
4. Use some combination of blurring and morphological operations (Sections 13.3.1 and 13.3.2) to remove very small regions in the difference image, and to fill in 'holes' etc. in larger ones. Surviving regions represent the moving objects.
5. Update the median measurement, incorporating frame $K + 1$ and discarding the current earliest measurement.
6. Return to (2) for the next frame.

Step 5 here is potentially very expensive, demanding the storage of the last K images and a per-pixel sort at each frame. An efficient short cut is available [McFarlane and Schofield, 1995]: only the current background is stored, and intensities are incremented by one if the current frame is brighter at the pixel, and decremented if it is darker. This trick converges the background onto an intensity for which half the updates are brighter and half darker—that is, the median.

This median filter approach to background maintenance is simple to understand and has been applied with some success. Figure 16.16 illustrates the result from a cheap, free-standing camera watching an unconstrained scene [Baumberg and Hogg, 1994a]. We will derive a large number of silhouettes that can be analyzed for shape—in this applica-



Figure 16.16: Left—a pedestrian scene; right—movement extracted from the scene by maintaining a median-filtered background. Post-processing has been applied to remove 'small' blobs and to tidy up the figure boundary. *Courtesy of A. M. Baumberg, University of Leeds.*

tion, each silhouette is characterized by a cubic B-spline (Section 8.2.5) determined by 40 boundary points (i.e., 80 parameters in 2D); these are then used to train a point distribution model (Section 10.4) which permits very efficient description of walking figures in 18 dimensions (the remaining 62 containing very little information).

Median-based background generation is straightforward and can be made to run fast. Nevertheless, it is limited in its application, being vulnerable to scenes in which there are many moving objects, or objects moving slowly. The choice of the interval length K in Algorithm 16.5.1, and the various parameters of difference thresholding and post-processing, make it sensitive. A more robust algorithm [Stauffer and Grimson, 1999] that compensates for these drawbacks is popularly used when the simple one fails.

The idea is to model each pixel independently, in each case as a mixture of Gaussians (see Section 10.13). Depending on local behavior, some of these Gaussians will represent background and some foreground: the algorithm provides a means of deciding which. This seemingly elaborate approach becomes necessary since:

- If each pixel were the result of a particular surface under invariant lighting, a single Gaussian would be sufficient to model its appearance under system noise.
- If the lighting were varying slowly over time, then a single adapting Gaussian would be sufficient.

In practice, multiple surfaces might well appear, and lighting conditions can change in a number of ways at a variety of rates.

Consider a particular pixel with intensity g_t at time t , $t = 1, 2, \dots$ (we consider here the simple case of gray-scale images—the formal algorithm will be presented for RGB). The recent history of the process g_t will be modeled as K Gaussians, $N_k = N(\mu_k, \sigma_k^2)$ where $k = 1, \dots, K$. We shall expect these Gaussians to evolve in time as the environment evolves, and objects come and go, so more properly we write

$$N_{kt} = N(\mu_{kt}, \sigma_{kt}^2) \quad k = 1, \dots, K. \quad (16.50)$$

The choice of K is governed by issues of computational efficiency (remember this is being executed at each pixel), and values in the range 3–7 are commonly used. Obviously, $K > 2$; otherwise, the idea reduces to a simple foreground-background model— $K = 3$ permits two backgrounds and one foreground model.

Associated with each Gaussian is a weight ω_{kt} which also evolves in time. Then the probability of observing g_t is

$$P(g_t) = \sum_{k=1}^K \omega_{kt} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(g_t - \mu_{kt})^2}{\sigma_{kt}^2}\right). \quad (16.51)$$

These weights are normalized to sum to 1.

As the process proceeds, we could in principle use the EM algorithm (see Section 10.13) to update the Gaussian parameters but this would prove very costly. Instead, the pixel is compared to each Gaussian: if it is within 2.5 standard deviations of the mean it is considered a ‘match’; if there is more than one match, the best such is taken—this is a ‘winner takes all’ strategy. Now:

- If a match is found, for Gaussian l say, we set

$$\begin{aligned} \omega_{kt} &= (1 - \alpha) \omega_{k(t-1)} & \text{for } k \neq l, \\ &= \omega_{k(t-1)} & \text{for } k = l, \end{aligned} \quad (16.52)$$

and then re-normalize the ω . α is a learning constant: $1/\alpha$ determines the speed at which parameters change. The parameters of the matched Gaussian are updated as

$$\begin{aligned}\mu_{lt} &= (1 - \rho) \mu_{l(t-1)} + \rho g_t, \\ \sigma_{lt}^2 &= (1 - \rho) \sigma_{l(t-1)}^2 + \rho (g_t - \mu_{lt})^2,\end{aligned}$$

where

$$\rho = \alpha P(g_t | \mu_l, \sigma_l^2).$$

- If a match is not found, the least popular Gaussian (lowest ω) is lost, and is replaced by a new one with mean g_t . It is assigned a high variance and low weight (relative to the other $K - 1$ distributions) at this stage. This is the mechanism whereby new objects are ‘spotted’, and gives them the opportunity, should they persist, of becoming part of the local background.

At this stage, the Gaussian most likely to have given the pixel its current intensity is known, and it remains to decide whether it is background or foreground. This is achieved via a constant T of the whole observation operation: it is assumed that in all frames, the proportion of background pixels always exceeds T . Then, the Gaussians are ordered on the expression ω_{kt}/σ_{kt} —a high value implies either a high weight, or a low variance (or both). Either of these conditions would encourage our belief that the pixel was background. Then the distributions $k = 1, \dots, B$ are considered to be background where

$$B = \operatorname{argmin}_b \left(\sum_{k=1}^b \omega_{kt} > T \right) \quad (16.53)$$

and thus a decision is given on the current pixel.

Formally, considering multi-dimensional pixels, the algorithm is:

Algorithm 16.6: Background maintenance by Gaussian mixtures

1. Initialize: Choose K the number of Gaussians and a learning constant α : values in the range 0.01–0.1 are commonly used. At each pixel, initialize K Gaussians $N_k = N(\boldsymbol{\mu}_k, \Sigma_k)$ with mean vector $\boldsymbol{\mu}_k$ and covariance matrix Σ_k , and corresponding weights ω_k . Since the algorithm will evolve this may safely be done crudely on the understanding that early measurements may be unreliable.
2. Acquire frame t , with intensity vector $\mathbf{x}_t$ —probably this will be an RGB vector $\mathbf{x}_t = (r_t, g_t, b_t)$. Determine which Gaussians match this observation, and select the ‘best’ of these as l . In the 1D case, we would expect an observation to be within, say, 2.5σ of the mean. In the multi-dimensional case, a simplifying assumption is made for computational complexity reasons: the different components of the observation are taken to be independent and of equal variance σ_k^2 , allowing a quick test for ‘acceptability’.
3. If a match is found as Gaussian l :
 - (a) Set the weights according to equation (16.52), and re-normalize.

(b) Set

$$\rho = \alpha N(\mathbf{x}_t | \boldsymbol{\mu}_l, \sigma_l)$$

and

$$\begin{aligned}\boldsymbol{\mu}_{lt} &= (1 - \rho) \boldsymbol{\mu}_{l(t-1)} + \rho \mathbf{x}_t, \\ \sigma_{lt}^2 &= (1 - \rho) \sigma_{l(t-1)}^2 + \rho (\mathbf{x}_t - \boldsymbol{\mu}_{lt})^T (\mathbf{x}_t - \boldsymbol{\mu}_{lt}).\end{aligned}$$

4. If no Gaussian matched $\mathbf{x}_t$: then determine $l = \underset{k}{\operatorname{argmin}}(\omega_k)$ and delete N_l . Then set

$$\begin{aligned}\boldsymbol{\mu}_{lt} &= \mathbf{x}_t, \\ \sigma_{lt}^2 &= 2 \max_k \sigma_{k(t-1)}^2, \\ \omega_{lt} &= 0.5 \min_k \omega_{k(t-1)}.\end{aligned}$$

(The algorithm is reasonably robust to these choices.)

5. Determine B as in equation (16.53), and thence from the current ‘best match’ Gaussian whether the pixel is likely to be foreground or background.
6. Use some combination of blurring and morphological dilations and erosions to remove very small regions in the difference image, and to fill in ‘holes’ etc. in larger ones. Surviving regions represent the moving objects in the scene.
7. Return to (2) for the next frame.

Figure 16.17 illustrates the evolving balance of Gaussians weights in a simple scene.

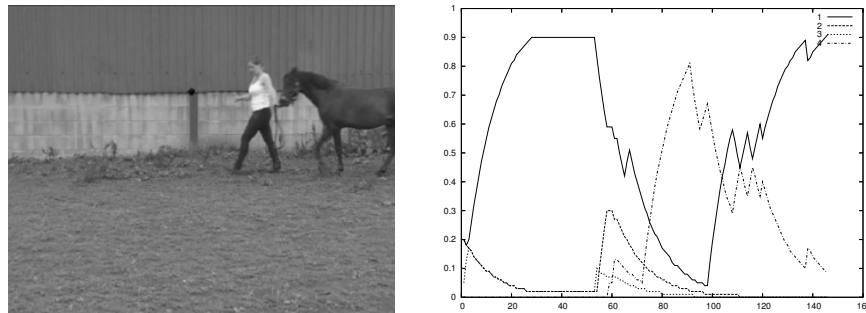


Figure 16.17: Progress of the Stauffer-Grimson background maintenance algorithm. During the sequence, the woman walks from right to left leading the horse; the indicated pixel (black dot, upper center) is background, is then obscured by her clothing and arm, and then the horse. The plot shows the weights of four Gaussians: at initialization the background, Gaussian 1, comes to dominate (capped here at weight 0.9). Other intensity patterns come and go (the horse is Gaussian 4), and at the end of the sequence the background resumes dominance. *Courtesy of B. S. Boyle, University of Leeds.*

This algorithm has proved very popular and has been the subject of many refinements: it is usefully and clearly considered from the point of view of implementation in [Power and Schoonees, 2002]. Particular useful enhancements are to allow α to start high (maybe 0.1) and reduce in time (perhaps to 0.01). Improvements to the Gaussian mixture models approach that simultaneously model multiple foregrounds have been very successfully demonstrated in tracking urban traffic scenes [Magee, 2004].

While other approaches exist (for example, the task may be performed using a PCA approach in which *eigen-backgrounds* are derived from a training set, and ‘foreground’ reveals itself as significant deviations from a mean image [Oliver et al., 1999]) most current implementations will use a variant of one of the algorithms presented here.

16.5.2 Kernel-based tracking

While background modeling may simplify object tracking, it can be more valuable not to require any specific background treatment: the requirement that only small changes occur between two consecutive frames is satisfied in most applications due to the high frame rate of video sequences. If this requirement is met, gradient-based object localization and tracking can be accomplished using a general correlation criterion [Bascle and Deriche, 1995] and considering illumination and geometry changes [Hager and Belhumeur, 1996]. An intriguing, highly efficient approach to real-time tracking was introduced in [Comaniciu et al., 2000]. Their method is based on spatially masking the target object with an isotropic kernel, followed by applying a smooth similarity function, which reduces the tracking problem to a maximum similarity search in the vicinity of the previous location. The similarity optimization is then efficiently performed using the mean shift algorithm (Section 7.1). An in-depth treatment of kernel-based object tracking is given in [Comaniciu et al., 2003].

First, the desired *target properties* must be determined—for example by estimating a probability density function q from image data. In real-world video tracking, working with color information is typical and information about the *color* distribution of the tracked object may be used to form a feature space. To track the target from frame to frame, the target model identified in the previous frame is first centered at the origin of a local coordinate system. The current-frame *target candidate* is then located at a position $\mathbf{y}$. The target candidate feature description can be characterized by a probability density function $p(\mathbf{y})$, which is estimated from the current image frame data. For computational efficiency, discrete probability density functions are employed; m -bin histograms are used as simple and frequently sufficient representations. As such, the target model $\hat{\mathbf{q}}$ and target candidate $\hat{\mathbf{p}}(\mathbf{y})$ probability density functions are defined as:

$$\hat{\mathbf{q}} = \{\hat{q}_u\}, \quad \sum_{u=1}^m \hat{q}_u = 1, \quad (16.54)$$

$$\hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}, \quad \sum_{u=1}^m \hat{p}_u = 1, \quad (16.55)$$

where $u = 1, \dots, m$. Let $\hat{\rho}(\mathbf{y})$ be a similarity function between $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$

$$\hat{\rho}(\mathbf{y}) \equiv \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]. \quad (16.56)$$

For a tracking task, the similarity $\hat{\rho}(\mathbf{y})$ coincides with the likelihood that a tracked target already located in the previous frame is present at position $\mathbf{y}$ in the current frame.

Therefore, the local maxima of $\hat{\rho}(\mathbf{y})$ correspond to the presence of the target in the current frame of the analyzed image sequence.

There are many ways the similarity function can be defined. Clearly, using solely spectral information cannot provide a smooth similarity function since even closely adjacent locations may exhibit large variability. Consequently, hill-climbing optimization techniques are insufficient, and exhaustive (and thus expensive) optimization techniques do not allow efficient implementations. One of the main ideas to achieve regularization of the similarity function is spatially masking the object with an isotropic kernel [Comaniciu et al., 2000]. The feature-space description is then represented by the kernel weights and $\hat{\rho}(\mathbf{y})$ is a smooth function in $\mathbf{y}$.

The *target model* is derived from an elliptic region that is first normalized to a unit circle to remove the influence of the target scale. The target region, consisting of n pixels, is thus represented using *normalized* pixel coordinates $\{\mathbf{x}_i^*\}$; the normalization uses the center of the unit circle as the origin. This target region is registered with a convex and monotonically decreasing kernel K with a profile $k(x)$ such that

$$k(x) : [0, \infty] \rightarrow \mathcal{R} \quad \text{so that} \quad K(\mathbf{x}) = k(\|\mathbf{x}\|^2). \quad (16.57)$$

Comaniciu et al. recommend using an Epanechnikov kernel (see Section 7.1 and equation 7.4). The kernel weights are in agreement with our understanding that pixels farther away from the center are less reliable due to possible occlusions, interference with background, or similar boundary effects. Recall that the image features are represented using an m -bin histogram. Each target model pixel at location $\mathbf{x}_i^*$ must be associated with the index $b(\mathbf{x}_i^*)$ of its bin in the quantized feature space using function $b : \mathcal{R}^2 \rightarrow \{1, \dots, m\}$. The probability $\hat{q}_u$ of the feature $u \in \{1, \dots, m\}$ is then computed as

$$\hat{q}_u = C \sum_{i=1}^n k(\|\mathbf{x}_i^*\|^2) \delta(b(\mathbf{x}_i^*) - u), \quad (16.58)$$

where δ is the Kronecker delta function² and C is a normalization constant

$$C = \frac{1}{\sum_{i=1}^n k(\|\mathbf{x}_i^*\|^2)}. \quad (16.59)$$

The *target candidate* is represented by *normalized* pixel locations $\{\mathbf{x}_i\}$ for $i = 1, \dots, n_h$, centered at $\mathbf{y}$ in the current frame, where h denotes the bandwidth of kernel K with the same profile $k(x)$ that was used for the target model. Importantly, the normalization is inherited from the frame containing the target model. The bandwidth defines the scale of the target candidate, thus also determining the number of pixels in the current frame that need to be analyzed during the localization process. The probability $\hat{p}_u$ of feature $u \in \{1, \dots, m\}$ is

$$\hat{p}_u = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right) \delta(b(\mathbf{x}_i) - u) \quad (16.60)$$

with the normalization constant

$$C_h = \frac{1}{\sum_{i=1}^{n_h} k(\|\mathbf{y} - \mathbf{x}_i\|^2/h^2)}. \quad (16.61)$$

²Kronecker delta: $\delta(l) = 1$ for $l = 0$; $\delta(l) = 0$ otherwise.

Since C_h does not depend on $\mathbf{y}$, it can be pre-calculated for the given kernels and bandwidths. The similarity function (equation 16.56) inherits the properties of the employed kernel K . Using a smooth differentiable kernel allows the similarity function optimization to use simple hill climbing techniques. Several applicable histogram similarity measures are compared in [Puzicha et al., 1999].

Naturally, the similarity function needs to be a *metric* to evaluate the *distance* between the target model and the respective candidates. The distance $d(\mathbf{y})$ between two distributions can be estimated using the Bhattacharyya coefficient assessing the similarity between $\mathbf{p}$ and $\mathbf{q}$ [Djouadi et al., 1990]:

$$d(\mathbf{y}) = \sqrt{1 - \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]} , \quad (16.62)$$

where

$$\hat{\rho}(\mathbf{y}) \equiv \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] = \sum_{u=1}^m \sqrt{\hat{p}_u(\mathbf{y}) \hat{q}_u} . \quad (16.63)$$

To find the most likely location of the target in the current frame, the distance (equation 16.62) must be minimized and/or the Bhattacharyya coefficient (equation 16.63) maximized as a function of $\mathbf{y}$. The optimization starts from the target model position in the previous frame and—relying on the smoothness property of the similarity function—uses the gradient-based mean shift approach (Section 7.1). Clearly, the current-frame target position must be within the attraction area of the similarity function (defined by the bandwidth of the kernel) for the hill-climbing to be successful.

Individual tracking steps start in the current frame from the target model position $\mathbf{y}_0$ determined in the previous frame. The model itself is first estimated from the initial frame of the tracked sequence. Due to possible changes in appearance over time, a mechanism to update the target model must be available. To initialize each tracking step, the probability of the target candidate $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}$ for $u = 1, \dots, m$ is computed at location $\mathbf{y}_0$. A Taylor expansion around the values $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}$ yields the following approximation of the Bhattacharyya coefficient (equation 16.63)

$$\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] \approx \frac{1}{2} \sum_{u=1}^m \sqrt{\hat{p}_u(\hat{\mathbf{y}}_0) \hat{q}_u} + \frac{1}{2} \sum_{u=1}^m \hat{p}_u(\mathbf{y}) \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} . \quad (16.64)$$

A reasonably close approximation results as long as the target candidate $\{\hat{p}_u(\hat{\mathbf{y}})\}$ does not change dramatically from the initial $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}$ in subsequent frames. Note that the requirement of $\{\hat{p}_u(\hat{\mathbf{y}}_0)\} > 0$ (or greater than some small ϵ) is enforceable for all $u = 1, \dots, m$ by excluding the violating features.

The tracking process optimizes the target candidate location as given in equation (16.60). By employing equation (16.64), the second right-hand side term of the following equation must be maximized with the first term being independent of $\mathbf{y}$

$$\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] \approx \frac{1}{2} \sum_{u=1}^m \sqrt{\hat{p}_u(\hat{\mathbf{y}}_0) \hat{q}_u} + \frac{C_h}{2} \sum_{u=1}^{n_h} w_i k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right) , \quad (16.65)$$

where

$$w_i = \sum_{u=1}^m \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \delta(b(\mathbf{x}_i) - u) . \quad (16.66)$$

The second term that is maximized reflects a density estimate computed with kernel profile $k(x)$ at $\mathbf{y}$ in the current frame, weighted by w_i . Using a mean shift procedure, the maximum can be efficiently located in a recursive fashion starting from location $\hat{\mathbf{y}}_0$ as follows (see also Figure 7.1 and equation 7.12)

$$\hat{\mathbf{y}}_1 = \sum_{i=1}^{n_h} \mathbf{x}_i w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right) / \sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right), \quad (16.67)$$

where $g(x) = -k'(x)$ is differentiable for $x \in [0, \infty)$ except at a finite number of points.

Algorithm 16.7: Kernel-based object tracking

1. Assumptions: The target model $\{\hat{q}_u\}$ exists for all $u = 1, \dots, m$. The tracked object location in the previous frame $\hat{\mathbf{y}}_0$ is known.
2. Using the previous frame target location $\hat{\mathbf{y}}_0$ as the initial location of the target candidate in the current frame, compute $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}$ for $u = 1, \dots, m$ and

$$\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_0), \hat{\mathbf{q}}] = \sum_{u=1}^m \sqrt{\hat{p}_u(\hat{\mathbf{y}}_0) \hat{q}_u}.$$

3. Derive weights $\{w_i\}$ for $i = 1, \dots, n_h$ according to equation (16.66).
4. Determine the new location of the target candidate according to equation (16.67).
5. Compute the new likelihood value $\{\hat{p}_u(\hat{\mathbf{y}}_1)\}$ for $u = 1, \dots, m$ and determine

$$\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_1), \hat{\mathbf{q}}] = \sum_{u=1}^m \sqrt{\hat{p}_u(\hat{\mathbf{y}}_1) \hat{q}_u}.$$

6. If the similarity between the new target region and the target model is less than that between the old target region and the model

$$\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_1), \hat{\mathbf{q}}] < \rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_0), \hat{\mathbf{q}}]$$

perform the remainder of this step: move the target region half way between the new and old locations

$$\hat{\mathbf{y}}_1 := \frac{1}{2}(\hat{\mathbf{y}}_0 + \hat{\mathbf{y}}_1), \quad (16.68)$$

and evaluate the similarity function in this new location

$$\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_1), \hat{\mathbf{q}}].$$

Return to the beginning of this step.

7. If $\|\hat{\mathbf{y}}_1 - \hat{\mathbf{y}}_0\| < \epsilon$, stop. Otherwise, use the current target location as a start for the new iteration, i.e., $\hat{\mathbf{y}}_0 := \hat{\mathbf{y}}_1$, and continue with step 3.

The value of ϵ in step 7 is chosen so that the vectors $\hat{\mathbf{y}}_0$ and $\hat{\mathbf{y}}_1$ would be referencing the same pixel in the original image coordinates. Usually, the maximum number of iterations is also limited to satisfy real-time performance requirements. Note that step 6 is only included to avoid potential numerical problems of mean shift maximization, which is a rare event, and in practice this step may be omitted. Consequently, the calculation of the Bhattacharyya coefficient is avoided in steps 2 and 5, yielding an additional speed-up for such a modification. Then, the algorithm only performs the weight computations in step 3, derives the new position in step 4, and tests the kernel shift in step 7. In that case, the Bhattacharyya coefficient is only computed after the convergence to evaluate the similarity between the target model and the candidate.

To facilitate changes of scale, the bandwidth h of the kernel must be properly adjusted during the tracking process. Let h_{prev} be the bandwidth used in the previous frame. The best bandwidth h_{opt} for the current frame is determined by repeating the target localization algorithm for three values of h :

$$h = h_{\text{prev}}, \quad (16.69)$$

$$h = h_{\text{prev}} + \Delta h, \quad (16.70)$$

$$h = h_{\text{prev}} - \Delta h, \quad (16.71)$$

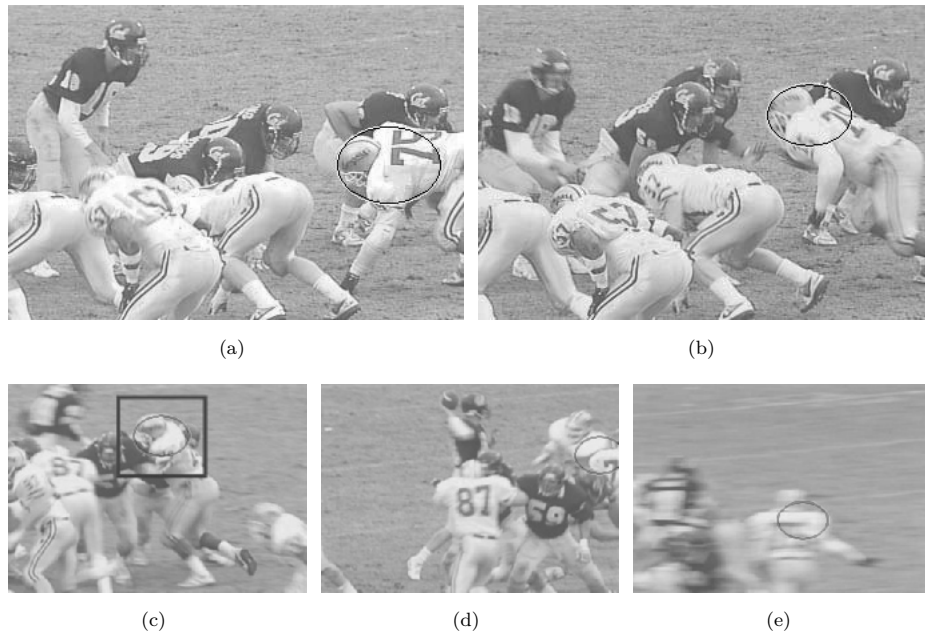


Figure 16.18: Kernel-based tracking of player #75: frames 30, 75, 105, 140, and 150 in a 154 long sequence. The tracker copes well with partial occlusion, camera motion, clutter, and blurring. Rectangular subwindow marked in panel (c) is used in Figure 16.19. *Courtesy of P. Meer, Rutgers University.* © 2003 IEEE. Reprinted, with permission, from D. Comaniciu, V. Ramesh, P. Meer, “Kernel-based object tracking,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, pp. 564-575, 2003.

with a 10% difference between the tested values being typical: $\Delta h = 0.1 h_{\text{prev}}$. The best bandwidth is determined by the highest value of the Bhattacharyya coefficient. To avoid an overly sensitive modifications of the bandwidth, the new bandwidth is determined as

$$h_{\text{new}} = \gamma h_{\text{opt}} + (1 - \gamma) h_{\text{prev}}, \quad (16.72)$$

typically $\gamma = 0.1$ is used. Clearly, the optimal bandwidth value as a function of time carries potentially valuable information about the tracked object.

Figure 16.18 shows 5 351×240 pixel frames from a 154-frame sequence; the goal was to track movement of player number 75. Target initialization was performed manually by drawing an ellipse as shown in panel (a). The color space was quantized in $16 \times 16 \times 16$ bins, which served as features. Figure 16.19a shows the number of mean shift iterations as a function of the frame number—about 4 iterations were needed on average for frame-to-frame tracking. Values of the Bhattacharyya similarity coefficient corresponding to Figure 16.18c are shown in Figure 16.19b together with the start- and end-locations of the mean shift iterations. Additional examples can be found in [Comaniciu et al., 2000, 2003].

The tracking task shown in Figure 16.18 did not assume any motion model. As such, the tracker was well suited to the abrupt changes of motion direction and unpredictable camera view changes typical of a football game video. However, background modeling (Section 16.5.1) and motion models (Section 16.6) can be incorporated in the kernel-based trackers, see [Comaniciu et al., 2003] for details. It is also worth noting that the kernel based trackers can track multiple objects in parallel, simply by maintaining several target models and their iterative tracking. The number of simultaneously tracked objects is only limited by the available computational resources—so that the real-time behavior of the trackers is not compromised. For example, if a single-target tracker can track

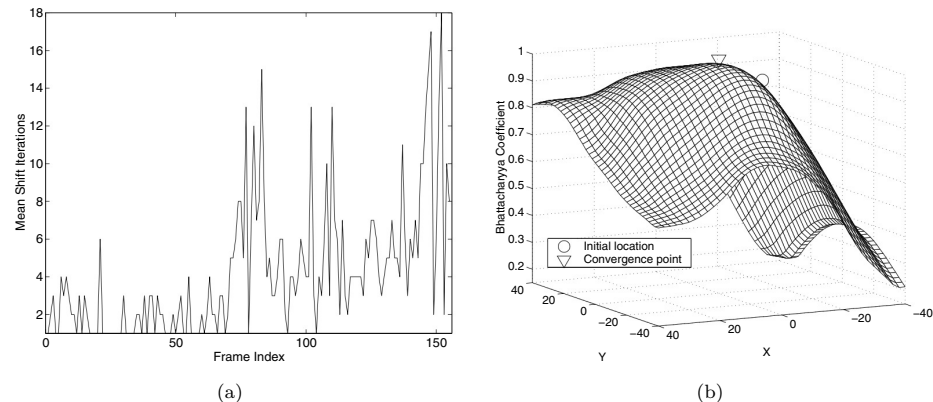


Figure 16.19: Performance of kernel-based tracking—see Figure 16.18. (a) Number of mean shift iterations as a function of frame number. (b) Similarity surface depicting the values of the Bhattacharyya coefficient within the rectangle shown in Figure 16.18c. Start and end locations of the mean shift iterations are depicted to show the optimization process yielding the object tracking behavior. *Courtesy of P. Meer, Rutgers University. © 2003 IEEE. Reprinted, with permission, from D. Comaniciu, V. Ramesh, P. Meer, “Kernel-based object tracking,” IEEE Trans. Pattern Anal. Machine Intell., vol. 25, pp. 564-575, 2003.*

objects at a frame rate of 250 frames per second, such a tracker can independently track 10 objects at a standard video frame rate of 25 fps.

16.5.3 Object path analysis

If there is only one object in the image sequence, the task can often be solved using approaches already described, but if there are many objects moving simultaneously and independently, more complex approaches are needed to incorporate individual object motion-based constraints. In such situations, motion assumptions/constraints described earlier should be examined (maximum velocity, small acceleration, common motion, mutual correspondence, smoothness of motion). Consequently, it is possible to formulate the notion of path coherence which implies that the motion of an object at any point in an image sequence will not change abruptly [Jain et al., 1995].

The **path coherence function** Φ represents a measure of agreement between the derived object trajectory and the motion constraints. Path coherence functions should follow four principles [Sethi and Jain, 1987; Jain et al., 1995]:

- The function value is always positive.
- The function reflects local absolute angular deviations of the trajectory.
- The function should respond equally to positive and negative velocity changes.
- The function should be normalized $[\Phi(\cdot) \in (0, 1]]$.

Let the trajectory T_i of an object i be represented by a sequence of points in the projection plane

$$T_i = (X_i^1, X_i^2, \dots, X_i^n), \quad (16.73)$$

where X_i^k represents a (three-dimensional) trajectory point in image k of the sequence (see Figure 16.20). Let $\mathbf{x}_i^k$ be the projection image co-ordinates associated with the point X_i^k . Then the trajectory can be expressed in vector form

$$T_i = (\mathbf{x}_i^1, \mathbf{x}_i^2, \dots, \mathbf{x}_i^n). \quad (16.74)$$

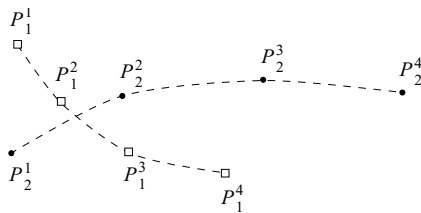


Figure 16.20: The trajectories of two objects moving simultaneously and independently.
© Cengage Learning 2015.

Deviation function

Deviation in the path can be used to measure path coherence. Let d_i^k be the deviation in the path of the point i in the image k

$$d_i^k = \Phi\left(\overline{\mathbf{x}_i^{k-1} \mathbf{x}_i^k}, \overline{\mathbf{x}_i^k \mathbf{x}_i^{k+1}}\right) \quad \text{or} \quad d_i^k = \Phi\left(X_i^{k-1}, X_i^k, X_i^{k+1}\right), \quad (16.75)$$

where $\overline{\mathbf{x}_i^{k-1} \mathbf{x}_i^k}$ represents the motion vector from point X_i^{k-1} to point X_i^k and Φ is the path coherence function. The deviation D_i of the entire trajectory of the object i is then

$$D_i = \sum_{k=2}^{n-1} d_i^k. \quad (16.76)$$

Similarly, for m trajectories of m moving objects in the image sequence, the overall trajectory deviation D can be determined as

$$D = \sum_{i=1}^m D_i. \quad (16.77)$$

With the overall trajectory deviation defined in this way, the multiple object trajectory tracking can be solved by minimizing the overall trajectory deviation D .

Path coherence function

It remains to define the path coherence function Φ . In agreement with motion assumptions, if the image acquisition frequency is high enough, the direction and velocity changes in consecutive images should be smooth. Then the path coherence function can be defined as

$$\begin{aligned} \Phi(P_i^{k-1}, P_i^k, P_i^{k+1}) &= w_1 \left(1 - \cos \theta \right) + w_2 \left(1 - 2 \frac{\sqrt{s_k s_{k+1}}}{s_k + s_{k+1}} \right) \\ &= w_1 \left(1 - \frac{|\overline{\mathbf{x}_i^{k-1} \mathbf{x}_i^k} \cdot \overline{\mathbf{x}_i^k \mathbf{x}_i^{k+1}}|}{\|\overline{\mathbf{x}_i^{k-1} \mathbf{x}_i^k}\| \|\overline{\mathbf{x}_i^k \mathbf{x}_i^{k+1}}\|} \right) + w_2 \left(1 - 2 \frac{\sqrt{\|\overline{\mathbf{x}_i^{k-1} \mathbf{x}_i^k}\| \|\overline{\mathbf{x}_i^k \mathbf{x}_i^{k+1}}\|}}{\|\overline{\mathbf{x}_i^{k-1} \mathbf{x}_i^k}\| \|\overline{\mathbf{x}_i^k \mathbf{x}_i^{k+1}}\|} \right), \end{aligned} \quad (16.78)$$

where the angle θ and distances s_k, s_{k+1} are given by Figure 16.21. The weights w_1, w_2 reflect the importance of direction coherence and velocity coherence.

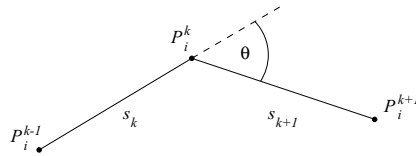


Figure 16.21: Path coherence function—definition of the angle θ and distances s_k, s_{k+1} . © Cengage Learning 2015.

Occlusion

When simultaneously tracking several objects with independent motion, object occlusion is almost guaranteed to occur. Consequently, some objects may partially or completely disappear in some image frames which can result in trajectory errors. If minimization of the overall trajectory deviation D (equation 16.77) is performed using the given path coherence function, it is assumed that the same number of objects (object points) is detected in each image and that the detected object points consistently represent the same objects (object points) in every image. Clearly, this is not the case if occlusion occurs.

To overcome this problem, additional local trajectory constraints must be considered and trajectories must be allowed to be incomplete if necessary. Incompleteness may reflect occlusion, appearance or disappearance of an object, or missing object points due to changed object aspect resulting from motion or simply due to poor object detection. Thus, additional motion assumptions that were not reflected in the definition of the path coherence function must be incorporated. An algorithm called the *greedy exchange* has been developed which finds the maximum set of complete or partially complete trajectories and minimizes the sum of local smoothness deviations for all identified trajectories. Local smoothness deviation is constrained not to exceed a preset maximum $\Phi_{\max}$ and the displacement between any two successive trajectory points X_i^k, X_i^{k+1} must be less than a preset threshold $d_{\max}$. To deal efficiently with incomplete trajectories, phantom points are introduced and used as substitutes for those which are missing. These hypothetical points allow each potential trajectory to be treated as complete and permit consistent

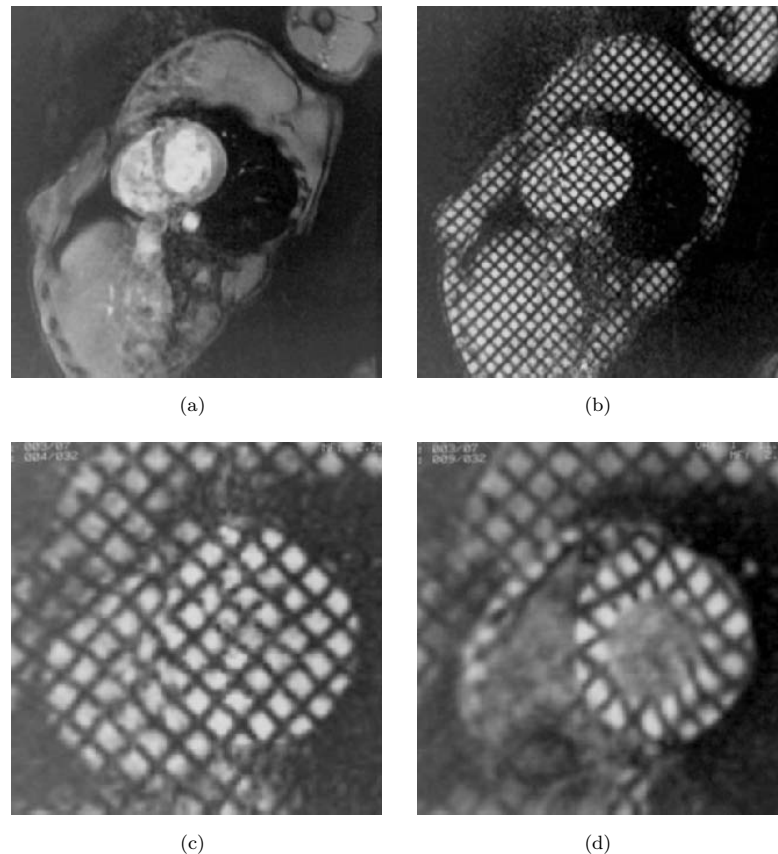


Figure 16.22: Magnetic resonance heart image. (a) Original chest image, diastole. (b) Chest image with magnetic resonance markers, diastole. (c) Image of the heart with markers, diastole. (d) Image with markers, systole. *Courtesy of D. Fisher, S. M. Collins, The University of Iowa.*

application of the optimization function. Details of the algorithm and example results can be found in [Sethi and Jain, 1987; Jain et al., 1995].

A conceptually similar method that minimizes a proximal uniformity cost function (reflecting the assumption that a small distance is usually traveled in a short time and a smooth trajectory is followed) was presented in [Rangarajan and Shah, 1991]. A two-stage algorithm exists, in which the first stage performs a forward search that extends trajectories up to the current frame and the second stage is a rule-based backward correcting algorithm that corrects wrong correspondences introduced in the last few frames.

Spatio-temporal approaches to analysis of image sequences with multiple independently moving objects represent another alternative to motion analysis. A minimum description length (MDL) approach to motion analysis of long image sequences has been developed: the method first constructs a family of motion models, each model corresponding to some meaningful motion type—translation, rotation, their combination, etc. Using the motion description length, the principle of progressive perception from extension in time, and optimal modeling of a limited period of observations, the objects in the image sequences are segmented to determine when objects change their type of motion or when a new part of an object appears. If the motion information in two consecutive frames is ambiguous, it is resolved by minimizing the motion description length in a long image sequence. Examples and applications for stationary and moving observers are given in [Gu et al., 1996].

A different method of interest-point correspondence and trajectory detection has been used in the analysis of cardiac wall motion from magnetic resonance images [Fisher et al., 1991], where rigid body motion assumptions could not be used since the human heart changes its shape over the cardiac cycle. Interest points were magnetically applied to the heart muscle using a special magnetic resonance pulse sequence known as SPAMM (spatial modulation of magnetization). This results in an image with a rectangular grid of markers, see Figure 16.22; heart motion is clearly visible on images if markers are applied. The first step of the motion analysis algorithm is a precise automatic detection

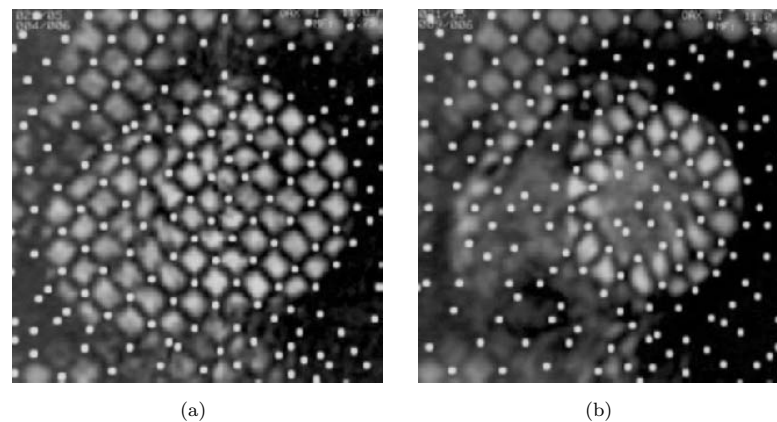


Figure 16.23: Detected position of markers. (a) Diastole. (b) Systole. *Courtesy of D. Fisher, S. M. Collins, The University of Iowa.*

of markers. Using a correlation technique (Section 6.4), the exact position of markers is determined (possibly at subpixel resolution), see Figure 16.23.

To track the marker locations, specific knowledge about small relative motion of marker positions in consecutive frames is used. Markers are considered as nodes of a two-dimensional graph, and dynamic programming (Section 6.2.5) is used to determine optimal trajectories. The optimality criterion is based on distance between markers in consecutive images, on the quality of marker detection, and on consistency of motion direction in consecutive images. Marker quality evaluation results from the marker detection correlation process. Successor nodes are determined by requiring that the trajectory length between successors be less than some specified constant. Identified and tracked markers are illustrated in Figure 16.24, and a velocity field is shown in Figure 16.25.

A survey of many other approaches to motion correspondence, trajectory parameterization, representation of relative motion and motion events, overview of useful region-based features, matching and classification approaches, and approaches to motion recognition including cyclic motion, lipreading, gesture interpretation, motion verb recognition, and temporal textures classification, is given in [Cedras and Shah, 1995].

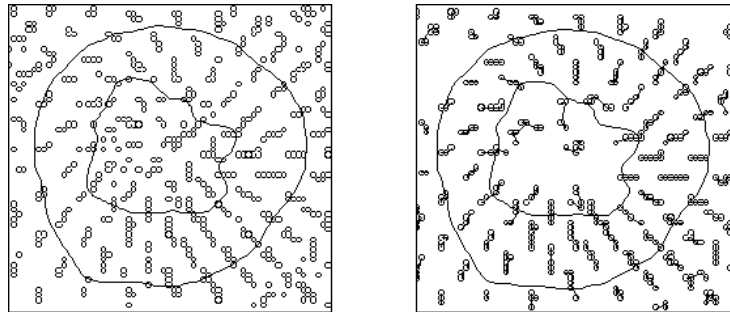


Figure 16.24: Velocity field. Identified markers (left) and tracked markers (right). Note that the dynamic programming has removed most of the spurious nodes that occur in the center of the cavity. *Courtesy of D. Fisher, S. M. Collins, The University of Iowa.*

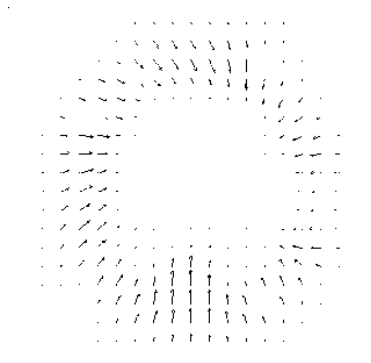


Figure 16.25: Velocity field derived from the information in Figure 16.24. *Courtesy of D. Fisher, S. M. Collins, The University of Iowa.*

16.6 Motion models to aid tracking

In fact, the tracking task represents an example of a very old and well-studied problem in control theory: the estimation of the state of a time-varying system via a sequence of noisy measurements. Here, the system states are the position, pose, etc. of the agents moving in the scene and the measurements are whatever feature vectors we choose to extract. This class of problem has been known since Gauss sought to determine the orbits of celestial bodies [Gauss, 1809], and it recurs in many fields of science and engineering.

Formally, we may have a model of an object in a scene that we see through noise; thus we *model* $\mathbf{x}$ and *observe* $\mathbf{z}$, where $\mathbf{x}$ and $\mathbf{z}$ are feature vectors, not necessarily of the same dimension. In the circumstance of observing through an image sequence, in particular motion, we may be able to use the models and observations in two ways.

- Multiple observations $\mathbf{z}_1, \mathbf{z}_2, \dots$ should permit an improved estimate of the underlying model $\mathbf{x}$. It is possible that this model evolves in time, in which case $\mathbf{z}_k$ will give an estimate of $\mathbf{x}_k$; provided we have a clear understanding of how $\mathbf{x}_k$ changes with k , it should still be possible to use the $\mathbf{z}_k$ to estimate this more complex model.
- The estimate of $\mathbf{x}$ at time k may also provide a prediction for the observation $\mathbf{x}_{k+1}$, and thereby for $\mathbf{z}_{k+1}$.

This suggests a feedback mechanism, or predictor controller (Figure 16.26), whereby we observe $\mathbf{z}_k$, estimate $\mathbf{x}_k$, predict $\mathbf{x}_{k+1}$ thereby predict $\mathbf{z}_{k+1}$, observe $\mathbf{z}_{k+1}$ *taking advantage of the prediction*, and then update our estimate of $\mathbf{x}_{k+1}$. This approach is in widespread use in computer vision, particularly within real-time tracking applications. The most popular ways of doing it are via **Kalman filters** [Kalman, 1960] or **particle filters**, particularly the Condensation algorithm [Isard and Blake, 1998]. Another increasingly popular approach to aid object tracking is to employ hidden Markov models (see Section 10.11).

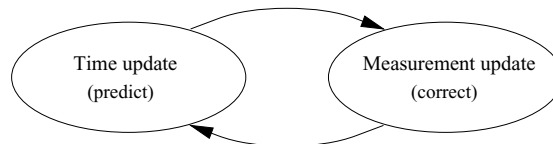


Figure 16.26: The predictor-corrector iterative cycle; the time update predicts events at the next step, and the measurement update adjusts estimates in the light of observation. © Cengage Learning 2015.

16.6.1 Kalman filters

For a particular class of model and observation, a classical approach delivers an optimal solution. The Kalman filter proceeds by assuming that the system is linear, that observations of it are linear functions of the underlying state, and that noise, both in the system and in measurement, is white and Gaussian (although slight variations and extensions of this model may be found in more specialized texts). In fact, the Kalman filter can be shown to provide a least-squares best estimate of system behavior—the model and observation estimates have minimal variance under the assumptions [Sorenson, 1970].

Formally³, we have the model

$$\begin{aligned}\mathbf{x}_{k+1} &= A_k \mathbf{x}_k + \mathbf{w}_k, \\ \mathbf{z}_k &= H_k \mathbf{x}_k + \mathbf{v}_k.\end{aligned}\tag{16.79}$$

The matrices A_k describe the evolution of the underlying model state, while $\mathbf{w}_k$ is zero mean Gaussian noise. We assume that $\mathbf{w}_k$ has covariance Q_k

$$\begin{aligned}Q_k &= E[\mathbf{w}_k \mathbf{w}_k^T], \\ (Q_k)_{ij} &= E(w_k^i w_k^j),\end{aligned}$$

where w_k^i denotes the i^{th} component of vector $\mathbf{w}_k$. The matrices H_k are the measurement matrices, describing how the observations are related to the model; $\mathbf{v}_k$ is another zero mean Gaussian noise factor, with covariance R_k .

Given $\mathbf{x}_{k-1}$ (or an estimate $\hat{\mathbf{x}}_{k-1}$) we can now compute from equation (16.79) an a priori estimate $\mathbf{x}_k = A_{k-1} \mathbf{x}_{k-1}$. Conventionally, this is referred to as $\hat{\mathbf{x}}_k^-$ to indicate (via $\wedge$) that it is an estimate, and (via $-$) that it is ‘before’ observation. Correspondingly, we can define $\hat{\mathbf{x}}_k^+$ as the updated estimate computed ‘after’ observation—it is this computation that the Kalman filter provides. Of course, we will be expecting $\hat{\mathbf{x}}_k^+$ to be an improvement on $\hat{\mathbf{x}}_k^-$.

Associated with each estimate are errors

$$\begin{aligned}\mathbf{e}_k^- &= \mathbf{x}_k - \hat{\mathbf{x}}_k^-, \\ \mathbf{e}_k^+ &= \mathbf{x}_k - \hat{\mathbf{x}}_k^+, \end{aligned}\tag{16.80}$$

with corresponding covariances P_k^- and P_k^+ . Note that these errors are caused by $\mathbf{w}_k$ and the error in the estimate.

The Kalman filter operates by examining the residual

$$\mathbf{z}_k - H_k \hat{\mathbf{x}}_k^-$$

to which $\mathbf{e}_k^-$ and the noise $\mathbf{v}_k$ contribute. In the absence of the noise, and if the estimate is perfect, this is zero. The approach is to seek a matrix K_k , the **Kalman gain** matrix, to update $\hat{\mathbf{x}}_k^-$ to $\hat{\mathbf{x}}_k^+$ in a least squares sense

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k (\mathbf{z}_k - H_k \hat{\mathbf{x}}_k^-).\tag{16.81}$$

If we can derive K_k , then $\hat{\mathbf{x}}_k^-$ may be updated to $\hat{\mathbf{x}}_k^+$ (since the other terms of the equation are known) and the problem is solved.

Equations (16.80) and (16.81) give

$$\begin{aligned}\mathbf{e}_k^+ &= \mathbf{x}_k - \hat{\mathbf{x}}_k^+ \\ &= \mathbf{x}_k - ((I - K_k H_k) \hat{\mathbf{x}}_k^- - K_k \mathbf{z}_k) \\ &= \mathbf{x}_k - (I - K_k H_k) \hat{\mathbf{x}}_k^- - K_k (H_k \mathbf{x}_k + \mathbf{v}_k) \\ &= (I - K_k H_k) \mathbf{e}_k^- + K_k \mathbf{v}_k.\end{aligned}\tag{16.82}$$

³The derivation that follows is non-trivial. For most readers, it may be sufficient to know that the assumptions of equation (16.79), and the results of equations (16.85)–(16.87) are sufficient to proceed to an implementation.

By definition

$$\begin{aligned} P_k^- &= E[\mathbf{e}_k^- \mathbf{e}_k^{-T}] , \\ P_k^+ &= E[\mathbf{e}_k^+ \mathbf{e}_k^{+T}] , \\ R_k &= E[\mathbf{v}_k \mathbf{v}_k^T] , \end{aligned}$$

and independence of errors gives

$$E[\mathbf{e}_k^- \mathbf{v}_k^T] = E[\mathbf{v}_k \mathbf{e}_k^{-T}] = 0 \quad (16.83)$$

and so equation (16.82) leads to

$$P_k^+ = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T . \quad (16.84)$$

K_k is now chosen to minimize the sum of the diagonal of P_k^+ , $\text{trace}(P_k^+)$, which is the sum of the posterior error variances—this is the ‘least squares’ aspect of the algorithm. This we do by taking the partial derivative of $\text{trace}(P_k^+)$ with respect to K_k and equating to zero. It is known (see, for example, [Gelb, 1974]) that if B is a symmetric matrix then

$$\frac{\partial}{\partial A} \text{trace}(ABA^T) = 2AB$$

and so from equation (16.84)

$$-2(I - K_k H_k) P_k^- H_k^T + 2K_k R_k = 0$$

whence, solving for K_k ,

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \quad (16.85)$$

wherein

$$P_k^- = A_k P_{k-1}^+ A_k^T + Q_{k-1} . \quad (16.86)$$

Elementary manipulation also provides the relation

$$P_k^+ = (I - K_k H_k) P_k^- . \quad (16.87)$$

If the functions describing the underlying state and/or the measurements are non-linear, an Extended Kalman filter can be obtained by linearization while the posterior density is still considered Gaussian [Bar-Shalom and Fortmann, 1988]. Parameterization of the mean and covariance of a posteriori density was introduced in [Julier and Uhlmann, 1997] and is called an **Unscented Kalman filter**.

Examples

In its formulaic form, the Kalman filter is not intuitive: to illustrate it, consider a trivial example [Gelb, 1974]. Suppose we have a constant scalar x , observed through uncorrelated Gaussian noise of zero mean and variance r . In this case we have $A_k = I = 1$ and $H_k = I = 1$:

$$\begin{aligned} x_{k+1} &= x_k , \\ z_k &= x_k + v_k , \end{aligned}$$

where v_k is normally distributed about 0 with variance r . Immediately, we see from equation (16.85) that

$$K_k = \frac{p_k^-}{p_k^- + r} . \quad (16.88)$$

We can deduce covariance relations from equations (16.86) and (16.87):

$$\begin{aligned} p_{k+1}^- &= p_k^+ , \\ p_{k+1}^+ &= (1 - K_{k+1}) p_{k+1}^- = \left(\frac{r}{p_{k+1}^- + r} \right) p_{k+1}^- = p_k^+ \frac{r}{p_k^+ + r} . \end{aligned} \quad (16.89)$$

Equation (16.89) provides a recurrence relation; writing $p_0 = p_0^+$, we can deduce

$$p_k^+ = \frac{r p_0}{k p_0 + r}$$

which, substituted into equation (16.88), gives

$$K_k = \frac{p_0}{r + k p_0}$$

and so, from equation (16.81),

$$\hat{x}_k^+ = \hat{x}_k^- + \frac{p_0}{r + k p_0} (z_k - \hat{x}_k^-) ,$$

which tells us the intuitively obvious—as k grows, new measurements provide less and less information.

More interestingly, consider the example introduced in Section 16.5.1 (page 816); recall that we have an 80D shape vector $\mathbf{x}$ (40 control points) which is reduced to an 18D representation via PCA:

$$\mathbf{x} = \bar{\mathbf{x}} + P \mathbf{b} ,$$

where $\mathbf{b}$ is the 18D vector derived from the PDM, and $\bar{\mathbf{x}}$ is the ‘mean shape’ (see Section 10.4). We note that a spline (object) appears in the scene after due translation, rotation, and scaling; if the current offset is (o_x, o_y) , the scale is s and the rotation is θ , we may model the boundary by

$$\begin{aligned} Q &= \begin{bmatrix} s \cos \theta & -s \sin \theta \\ s \sin \theta & s \cos \theta \end{bmatrix} , \\ \begin{bmatrix} X_i \\ Y_i \end{bmatrix} &= Q \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} o_x \\ o_y \end{bmatrix} , \end{aligned} \quad (16.90)$$

where x_i, y_i are the 40 two-dimensional points defining the spline. Then if we write $\mathbf{o} = (o_x, o_y, o_x, o_y, \dots, o_x, o_y)$ (40 times), and

$$\mathcal{Q} = \begin{bmatrix} Q & \dots & 0 \\ \vdots & \dots & \vdots \\ 0 & \dots & Q \end{bmatrix} \quad (16.91)$$

(an 80×80 matrix, which is 40 instances of the same 2×2 matrix), then the shape vector $\mathbf{X}$ is related to the state $\mathbf{b}$ by the equation

$$\mathbf{X} = \mathcal{Q}(P\mathbf{b} + \bar{\mathbf{x}}) + \mathbf{o}. \quad (16.92)$$

When a new object is detected, it will not be clear what the best estimates of its scale, trajectory, or model parameters are, but given suitable assumptions we might initialize these parameters and then iterate them during successive frames using a Kalman filter to converge on a good estimate [Baumberg and Hogg, 1994b]. We suppose that the object to be initialized has bounding box given by lower left co-ordinates (x_l, y_l) and upper right (x_r, y_r) , and that the mean height of figures in a training set is h_m . Rewriting equation (16.90) as

$$\begin{bmatrix} s \cos \theta & -s \sin \theta \\ s \sin \theta & s \cos \theta \end{bmatrix} = \begin{bmatrix} a_x & -a_y \\ a_y & a_x \end{bmatrix},$$

we can then initialize the figure as the mean shape, $\hat{\mathbf{b}}^0 = \mathbf{0}$, and

$$\begin{aligned} \hat{a}_x^0 &= \frac{y_r - y_l}{h_m}, & \hat{a}_y^0 &= 0, \\ \hat{o}_x^0 &= \frac{x_l + x_r}{2}, & \hat{o}_y^0 &= \frac{y_l + y_r}{2}, \end{aligned}$$

so the figure is scaled to its bounding box, aligned vertically, with the origin at the center of the box.

A stochastic model for the shape will be more stable than an assumption of uniform change, so we assume

$$\mathbf{b}^k = \mathbf{b}^{k-1} + \mathbf{w}^{k-1},$$

where $\mathbf{w}$ is a zero mean, normally distributed noise term, $w_i^k \sim N(0, \sigma_i)$. The eigen-analysis gives the variance of b_i over the training set as λ_i , and σ_i is initialized as $\sigma_i = \kappa \lambda_i$, where characteristically $\kappa = 0.05$ —thus the shape estimate is allowed to vary within an ellipsoid that is a subset of that defined by the training set. (Recall that the eigen-decomposition assumes that $E(b_i b_j) = 0$.)

Then we will assume that:

1. The object is moving uniformly in 2D subject to additive noise

$$\frac{d}{dt} \begin{bmatrix} o_x \\ o_y \end{bmatrix} = \begin{bmatrix} \dot{o}_x \\ 0 \end{bmatrix} + \begin{bmatrix} v_x \\ w_x \end{bmatrix},$$

for noise terms v_x and w_x (and similarly for o_y). This gives the frame update equation

$$\begin{bmatrix} o_x^{k+1} \\ \dot{o}_x^{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} o_x^k \\ \dot{o}_x^k \end{bmatrix} + \begin{bmatrix} v_x \\ w_x \end{bmatrix} \quad (16.93)$$

(where Δt is the time step between frames). v_x and w_x are noise terms, with $v_x \sim N(0, q_v)$, $w_x \sim N(0, q_w)$.

2. The alignment parameters a_x, a_y are constant, subject to noise

$$\begin{bmatrix} a_x^{k+1} \\ a_y^{k+1} \end{bmatrix} = \begin{bmatrix} a_x^k \\ a_y^k \end{bmatrix} + \begin{bmatrix} w_{ax} \\ w_{ay} \end{bmatrix}, \quad (16.94)$$

where $w_{ax}, w_{ay} \sim N(0, q_a)$.

The Kalman filter proceeds by estimating origin, alignment, and shape independently of each other. Given estimates of the parameters, we predict the next state $\hat{\mathbf{X}}^-$ from equation (16.92) and use this to make an observation from the image $\mathbf{z}$.

To update the x -origin co-ordinate we need to consider the state $(\hat{o}_x, \hat{o}_x)$. Many measurements of the origin are available from the expression

$$\mathbf{z} - \mathcal{Q}(P\hat{\mathbf{b}} + \bar{\mathbf{x}}).$$

These measurements, together with the update relation equation (16.93) and the noise variance properties, provide the necessary ingredients for the application of the filter, which can then be used to provide the best estimate of the origin in the next frame.

To update the alignment parameters with the observation $\mathbf{z}$, we use the measurement model

$$\mathbf{z} - \hat{\mathbf{o}} = H \begin{bmatrix} a_x \\ a_y \end{bmatrix},$$

where H is a $N \times 2$ measurement matrix defined by manipulation of equation (16.92). Again, updating estimates for the alignment is now a straightforward application of the theory.

Likewise, equation (16.92) provides a measurement model for the 18 shape parameters; it can be shown that each of these may be extracted independently of the others, permitting a model of the form

$$\mathbf{z} - \hat{\mathbf{X}} = \mathbf{h}_i (b_i - \hat{b}_i)$$

to be constructed, where $\mathbf{h}_i$ is an $N \times 1$ measurement matrix.

The Kalman filter is key to providing real-time performance in this application. Estimates of where the silhouette ought to be permit very localized search for edges in new frames, obviating the need for operations over the whole image. This operation is performed by determining the edge normal at the predicted position, and searching along it for the position of maximum contrast with the reference background. If no likely point is found (the contrast is low), ‘no observation’ is recorded and the associated predictions are not updated. This is particularly powerful since it permits the figure to suffer partial occlusion and still be tracked—this is illustrated by scenes with artificially introduced occlusion in Figure 16.27. This application is described in full detail, with various elaborations, in [Baumberg, 1995].



Figure 16.27: Tracking through occlusion. *Courtesy of A. M. Baumberg, University of Leeds.*

16.6.2 Particle filters

Kalman filters are an established piece of control theory and have proved very useful in aiding the tracking task. However, the underlying assumptions are limiting, and often restrict their use in real-world scenarios. Tracking through substantial noise and clutter in real-time (the requirement for many systems) is problematic: the assumption of local unimodal, Gaussian distributions is often invalid. Indeed, in many applications there will be no satisfactory approximation with an accessible mathematical representation. This has led to the application of a more general approach using **particle filters** in which systems are represented by sets of probabilistically derived samples that provide an empirical description of what is and is not 'likely'. Interestingly, this more general (and powerful) approach can be made to run satisfactorily in real-time, and has a more accessible analysis than the Kalman filter.

A particle filter approximates distributions by exploiting their temporal structure; in computer vision they were popularized mainly in the CONDitional DENSITY propagation—**CONDENSATION**—algorithm [Isard and Blake, 1998], on which this presentation is based. Supposing a system to be in states $X_t = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t\}$, where the subscript t denotes time, we may at time t have a probability density function telling us what $\mathbf{x}_t$ is likely to be. This will be represented by a set of **particles**—a set of sample states—whose occurrence will be governed by the p.d.f. As before, we will also have a sequence of observations $Z_t = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_t\}$ probabilistically related to the $\{\mathbf{x}_i\}$, and a Markovian assumption that $\mathbf{x}_t$ depends probabilistically on the preceding state $\mathbf{x}_{t-1}$, and that we can model this $P(\mathbf{x}_t|\mathbf{x}_{t-1})$. The important difference is that there is no constraint (in particular of linearity or Gaussian behavior) on any of these relations.

Condensation is an iterative procedure that maintains at each step a set of N samples $\mathbf{s}_i$ with associated weights π_i

$$\begin{aligned} S_t &= \{(\mathbf{s}_i, \pi_i)\}, \quad i = 1, 2, \dots, N, \quad \sum_i \pi_i = 1, \\ &= \{(\mathbf{s}_{ti}, \pi_{ti})\}. \end{aligned} \quad (16.95)$$

Together, these samples and weights represent the probability density function of $\mathbf{x}_t$ given Z_t ; this distribution is not expected to have a formulaic representation; in particular it may be multi-modal. The task is to derive S_t given S_{t-1} .

In order to generate N new samples and associated weights at time t , we select N times from S_{t-1} *taking account of the weights*, subject the sample to the Markov assumption (whatever it might be), and then re-weight the result in the light of the known observation $\mathbf{z}_t$. Formally:

Algorithm 16.8: Condensation (particle filtering)

1. Assume a weighted sample set at time $t - 1$ is known

$$S_{t-1} = \{(\mathbf{s}_{(t-1)i}, \pi_{(t-1)i})\}, \quad i = 1, 2, \dots, N.$$

Set

$$\begin{aligned} c_0 &= 0, \\ c_i &= c_{i-1} + \pi_{(t-1)i}, \quad i = 1, 2, \dots, N, \end{aligned} \quad (16.96)$$

(the cumulative probabilities).

2. To determine the n^{th} sample of S_t , select a random number in the range $[0,1]$, and determine $j = \text{argmin}_i(c_i > r)$; we shall propagate sample j . This is called *importance sampling*, a sampling technique that weights towards the more probable.
3. *Prediction* (Figure 16.26): Use knowledge of the Markovian behavior of $\mathbf{x}_t$ to derive $\mathbf{s}_{tn}$. How precisely this is done depends upon the Markov relationship: in the Kalman case, we would have

$$\mathbf{s}_{tn} = A_{t-1}\mathbf{s}_{(t-1)j} + \mathbf{w}_{t-1}$$

for matrices A_{t-1} and noise $\mathbf{w}_{t-1}$, but this relationship is not constrained. Importantly, note that $\mathbf{s}_{(t-1)j}$ may well be selected more than once as we iterate from (2), but this propagation may be expected to generate different $\mathbf{s}_{tn}$ as a result of the noise.

4. *Correction* (Figure 16.26): Use the current observation $\mathbf{z}_t$ and knowledge of the observation probabilities to set

$$\pi_{tn} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_{tn}) .$$

5. Iterate N times from (2).
6. Normalize $\{\pi_{ti}\}$ so that $\sum_i \pi_{ti} = 1$.
7. Our best estimate of $\mathbf{x}_t$ will now be

$$\mathbf{x}_t = \sum_{i=1}^N \pi_{ti} \mathbf{s}_{ti} \quad (16.97)$$

or, more generally for any moment

$$E[f(\mathbf{x}_t)] = \sum_{i=1}^N \pi_{ti} f(\mathbf{s}_{ti}) .$$

This algorithm can model arbitrarily complex p.d.f.'s, allowing it to maintain an arbitrary number of hypotheses simultaneously. Eventually we might expect the data evolution to reduce this to a smaller number until a single one survives.

A simple example might illustrate: suppose we have a 1D process so x_t and z_t are just real numbers. Suppose at time t that x_t is subject to a known displacement v_t (perhaps v_t is constant, or x_t exhibits simple harmonic motion), corrupted by zero-mean normally distributed noise

$$x_{t+1} = x_t + v_t + \epsilon_t , \quad \epsilon_t \text{ distributed as } N(0, \sigma_1^2)$$

and further suppose that z_t is observed through some blur, such that z may be expected to be normally distributed about x with variance σ_2^2 . Condensation would operate by

initializing N ‘guesses’ at x_1 , $S_1 = \{s_{11}, s_{12}, \dots, s_{1N}\}$; in the absence of any other information the initial weights may be uniform.

Now S_2 is generated: select s_j from S_1 by importance sampling (whatever the values of π_{1i}), and set $s_{21} = s_j + v_1 + \epsilon$, where ϵ is drawn from $N(0, \sigma_1^2)$ —repeat this N times to generate the particles for $t = 2$. Now set

$$\pi_{2i} = \exp\left(\frac{(s_{2i} - z_2)^2}{\sigma_2^2}\right)$$

then renormalize the π_{2i} and the iteration is complete. The best guess for x_2 is

$$\sum_{i=1}^N \pi_{2i} s_{2i}.$$

This example is trivial: in most applications, we would not expect $\mathbf{x}$ and $\mathbf{z}$ to belong to the same domain (as they do here): commonly, $\mathbf{x}$ might be a parameterized boundary (or part), while $\mathbf{z}$ might be a local pixel property, perhaps an intensity gradient [Isard and Blake, 1998]. $p(\mathbf{z}|\mathbf{x})$ will then measure the probability of the local observed pixel effects $\mathbf{z}$ given a certain parameter choice $\mathbf{x}$.

A more sophisticated example will illustrate better how this algorithm may be used in practice. Black and Jepson [Black and Jepson, 1998] consider the problem of recognizing gestures written on a whiteboard—this implies tracking the hand making the gesture and comparing its trajectory to a number of known models (there are nine, of varying complexity). A simplification of this application follows: one of these trajectories is a sequence of x, y velocities sampled uniformly in time;

$$\mathbf{m} = \{(\dot{x}_0, \dot{y}_0), (\dot{x}_1, \dot{y}_1), \dots, (\dot{x}_N, \dot{y}_N)\}.$$

(For clarity, we will omit the dots henceforward.) These models may be built in a number of ways—perhaps derived from the mean of a training set.

An input to the system at any instant will be a *partial* (as yet incomplete) trace—to match it we require to know:

- ϕ : The position (phase) within the model that aligns it with the current input; that is, how much of the model has been executed to date.
- α : An amplitude scaling factor, indicating how much taller (or shorter) the input is with respect to the model.
- ρ : A temporal scaling factor, indicating how much faster (or slower) the input is being provided with respect to the model.

A state of the system (which we will be propagating via Condensation) is then $\mathbf{s} = (\phi, \alpha, \rho)$: given an observed trajectory Z_t and a state $\mathbf{s}$, we can transform $Z_t = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_t)$ to be a partial model. We will seek to match these observations to a recent time window w wide.

The probability of this observation, given the state $\mathbf{s}$, will be

$$P(\mathbf{z}_t|\mathbf{s}) = P(z_t^x|\mathbf{s}) \times P(z_t^y|\mathbf{s}),$$

where

$$P(z_t^x | \mathbf{s}) = \frac{1}{\sqrt{2\pi} \sigma_x} \exp \left(-\frac{\sum_{j=0}^{w-1} (z_{t-j}^x - \alpha m_{\phi-\rho j}^x)^2}{2 \sigma_x (w-1)} \right)$$

and $P(z_t^y | \mathbf{s})$ similarly. σ_x and σ_y are estimates of the standard deviations of the relevant measurements.

1000 samples were initialized randomly with $\alpha \in [0.7, 1.3]$ and $\rho \in [0.7, 1.3]$. ϕ is initialized to be ‘small’ and the initial weights can be set to be uniformly $1/1000$. The Markovian relationships are

$$\begin{aligned} \phi_t &= \phi_{t-1} + \rho_{t-1} + \epsilon_\phi, & \epsilon_\phi &= N(0, \sigma_\phi), \\ \alpha_t &= \alpha_{t-1} + \epsilon_\alpha, & \epsilon_\alpha &= N(0, \sigma_\alpha), \\ \rho_t &= \rho_{t-1} + \epsilon_\rho, & \epsilon_\rho &= N(0, \sigma_\rho); \end{aligned}$$

$\sigma_\phi = \sigma_\alpha = \sigma_\rho = 0.1$ has been seen to work.

The Condensation algorithm has proved very popular in a number of examples—Figure 1.3 shows just one example. A number of enhancements have been developed: a particular issue has been the tracking of multiple objects, when the computational load of the algorithm starts to become prohibitive, and also independent trackers are prone to coalescing on the strongest evidence they find (the most evident object). Solutions to this problem have appeared as partitioned sampling [MacCormick and Blake, 1999]. Khan et al. [Khan et al., 2005] consider the interesting problem of tracking multiple objects that are interacting, using a Markov chain Monte Carlo [MCMC] approach, and this is enhanced by French [French, 2006] in modeling ‘co-operative’ agents, in this case flocking ducks. Interaction between the agents moving in the scene provides new information and constraints on the expected (relative) motion that can be used to advantage.

16.6.3 Semi-supervised tracking—TLD

The area of tracking has proved extremely fertile, both in application and in theoretical (and practical) depth. The appetite for real-time tracking is enormous and already the list of active areas is very long. For example, consider vision-based gesture recognition: this is an area in which techniques such as those discussed here are proving very fertile in assisting real-time interpretation of gesture by the user in unconstrained office scenes (that is, very cluttered); it should be easy to see how robust solutions to this problem would have very far-reaching effects on computer interfaces for users everywhere. (A snapshot of the state of this activity in 2006 is in [Gibet et al., 2006]).

While a number of powerful trackers have been developed, most are limited by problems surrounding training, appearance change and handling intermittent occlusion or disappearance. Any success implies the incorporation of a *detector* to spot the reappearance of an object, which in turn implies the pre-training of the detector. The detector would need adapt to object appearances not yet encountered, usually achieved by adaptive tracking—but if the tracker ‘loses’ the object, adaptive updates to the detector may worsen rather than improve matters.

Predator is a successful tracker designed to overcome these difficulties [Kalal et al., 2009] (from which this description is adapted) that simultaneously tracks, learns and detects: for this reason the framework is often referred to as **TLD**. Suppose we have an image sequence $I_0, I_1, \dots$, and are given the bounding box B_0 of an object at time $t = 0$:

we seek the bounding boxes of the object $B_1, B_2, \dots$ at subsequent times. The sequence B_t represents the trajectory of the object T_t —it is probable that we will describe the object in a feature space U , in which we will call the trajectory T_t^f . An unknown subset of $L^* \subset U$ represents all possible appearances of the object; at time $t = 0$ we know exactly one element $x_0 \in L^*$.

Now suppose a tracker is implemented (the Lucas-Kanade approach of Section 16.3.2 is usually used); for small values of t this will permit (probably) reasonable growth $L_1, L_2, \dots$ toward L^* , but if the tracker loses accuracy then L_t will start to incorporate poor or wrong instances of the model.

At a general time t we will expect L_t to include some correct model instances L_t^c , and some incorrect ones L_t^e ;

$$L_t = L_t^c \cup L_t^e, \quad L_t^c \subset L^*, \quad L_t^e \cap L^* = \emptyset$$

L_t is derived from L_{t-1} by two processes:

- **Growing:** the trajectory to date T_t^f is evaluated and the subset P that is considered to be positive identified; then we set $L_t = L_{t-1} \cup P$.
- **Pruning:** a subset N of L_{t-1} is estimated that contains incorrect instances of the model, and we set $L_t = L_{t-1} - N$.

The success or otherwise of this simple idea depends of course on the tracker, and the quality of the estimates of P and N .

The original implementation used the Lucas-Kanade tracker to estimate frame-to-frame bounding box movement and scale change, and represented the model by 15×15 intensity-normalized patches. We can then compute the normalized cross correlation between two patches as $NCC(x_i, x_j)$, and take the distance between them to be $d((x_i, x_j)) = 1 - NCC(x_i, x_j)$. The distance of a patch x_i from L_t is defined as

$$d(x_i, L_t) = \min_{x \in L_t} d(x_i, x).$$

The detector has exacting demands: it needs to be very fast, to learn fast, and to perform well on limited training data. Unlike other ‘dynamic’ algorithms that adapt to new instances and ‘forget’ older ones, it needs to retain all it has learned. The original approach used is a simplification of local binary patterns (Section 15.1.6) called a 2-bit binary classifier (2bitBP): each patch generates a number of such 2-bit responses and a random forest classifier is trained from them (see Section 9.9). These choices satisfy the requirements of the algorithm but are not critical to it.

Likewise, the precise implementation of growing and pruning need to be chosen. The first implementation did this by:

Growing: A threshold θ is chosen. If the distance of x_i to L_t is less than this threshold, and for some $j > i$, x_j is also less than θ to L_t , then all x_k , $i < k < j$ are added to the model. The reasoning is that if a tracker drifts off the model, it is unlikely to return; but if it does return, the deviation from the model is probably better explained by change in appearance of the tracked object.

Pruning: If the assumption is made that an object appears in the image at most once, then if the tracker and detector agree on object locations, all alternative detections are marked as false positives and pruned.

TLD is a framework or paradigm, and the choices itemized above pertain to the Predator instance (its first). At a high level, the algorithm is:

Algorithm 16.9: Tracking-Learning-Detection—TLD

1. Initialize a model with a correct instance x_0 : $L_0 = x_0$. (This may be done by a user outlining a bounding box in the first frame of a sequence.)
2. At time t , track x_{t-1} into the current frame.
3. Detect all instances of L_{t-1} in the current frame.
4. Determine the positive examples P and grow, and determine the negative examples N and prune: $L_t = L_{t-1} \cup P - N$.
5. Set x_t to be the most confident patch: this may be the result of tracking or detection.
6. Set $t = t + 1$ and go to 2.

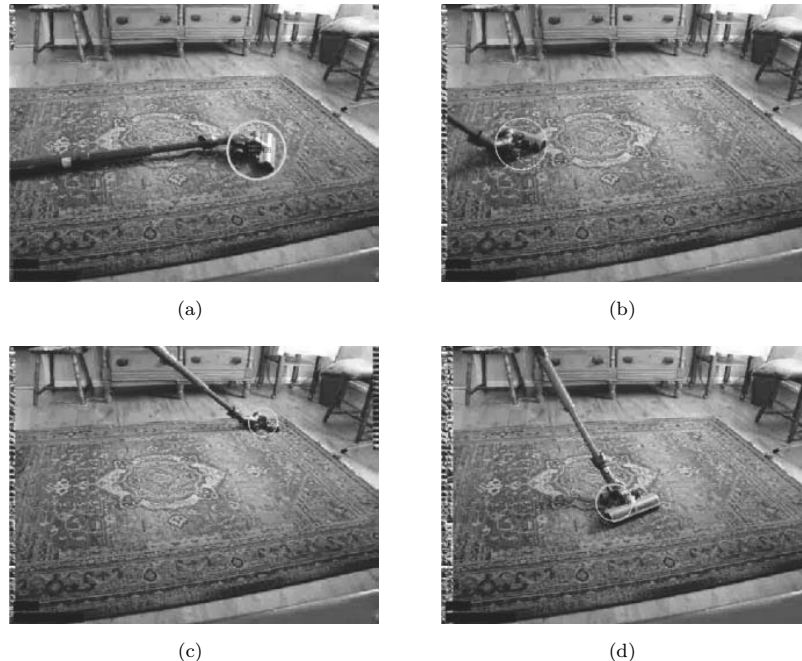


Figure 16.28: Predator: This simple example illustrates successful tracking of the head of a vacuum cleaner. (a) Initialization, done by hand; (b) Movement across a very challenging background (the rug), and change of pose; (c) Another change of pose and significant change of scale; (d) Further change of scale and significant change of pose—note only the leftmost part of the machine (that was originally visible) is being tracked. © Cengage Learning 2015.

The TLD resource page <http://info.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html> contains a good account of this algorithm, together with demonstration software and data, and source codes. An example, produced with this software, is shown in Figure 16.28; illustrations of the tracker surviving occlusion and disappearance of targets may be viewed online. TLD has received close study since its development, together with formal justification of its behavior—see, for example, [Kalal et al., 2010, 2012].

16.7 Summary

- **Motion analysis**

- Motion analysis is dealing with three main groups of motion-related problems:
 - * Motion detection
 - * Moving object detection and location
 - * Derivation of 3D object properties
- A two-dimensional representation of a (generally) three-dimensional motion is called a **motion field** wherein each point is assigned a **velocity vector** corresponding to the motion direction, velocity, and distance from an observer at an appropriate image location.
- Motion analysis and object tracking combine two separate but interrelated components:
 - * Localization and representation of the object of interest (target).
 - * Trajectory filtering and data association.
 - * One or the other may be more important based on the nature of the motion application.
- **Optical flow** represents one approach to motion field construction, in which motion direction and motion velocity are determined at possibly all image points.
- **Feature point correspondence** is another method for motion field construction. Velocity vectors are determined only for corresponding feature points.
- Object **motion parameters** can be derived from computed motion field vectors.
- **Motion assumptions** can help to localize moving objects. Frequently used assumptions include:
 - * Maximum velocity
 - * Small acceleration
 - * Common motion
 - * Mutual correspondence

- **Differential motion analysis**

- Subtraction of images acquired at different instants in time makes motion detection possible, assuming a stationary camera position and constant illumination.

- There are many problems associated with this approach, and results of subtraction are highly dependent on an object–background contrast.
- A **cumulative difference image** improves performance of differential motion analysis. It provides information about motion direction and other time-related motion properties, and about slow motion and small object motion.
- Detecting **moving edges** helps further overcome the limitations of differential motion analysis methods. By combining the spatial and temporal image gradients, the differential analysis can be reliably used for detection of slow-moving edges as well as detection of weak edges that move with higher speed.

- **Optical flow**

- Optical flow reflects the image changes due to motion during a time interval dt which must be short enough to guarantee small inter-frame motion changes.
- The optical flow field is the velocity field that represents the three-dimensional motion of object points across a two-dimensional image.
- Optical flow computation is based on two assumptions:
 - * The observed brightness of any object point is constant over time.
 - * Nearby points in the image plane move in a similar manner (the **velocity smoothness** constraint).
- Optical flow computation will be in error if the constant brightness and velocity smoothness assumptions are violated. In real imagery, their violation is quite common. Typically, the optical flow changes dramatically in highly textured regions, around moving boundaries, at depth discontinuities, etc. Resulting errors propagate across the entire optical flow solution.
- Global error propagation is the biggest problem of global optical flow computation schemes, and local optical flow estimation helps overcome the difficulties.
- Optical flow analysis does not result in motion trajectories; instead, more general motion properties are detected that can significantly increase the reliability of complex motion analysis. Parameters that are detected include:
 - * Mutual object velocity
 - * Focus of expansion (FOE) determination
 - * Distance (depth) determination
 - * Collision prediction

- **Motion analysis based on correspondence of interest points**

- This method finds significant points (**interest points, feature points**) in all images of the sequence—points least similar to their surroundings, representing object corners, borders, or any other characteristic features in an image that can be tracked over time.
- The Lucas-Kanade tracker is widely used: it tracks robustly by solving a linear system, assuming local patches move similarly.
- The KLT tracker uses the Lucas-Kanade approach to derive *automatically* points that are robust through an image [video] sequence.

- Point detection is followed by a matching procedure, which looks for correspondences between these points in time.
- The process results in a sparse velocity field.
- Motion detection based on correspondence works even for relatively long inter-frame time intervals.

- **Detection of specific motion patterns**

- Motion-specific information can be derived from training sets of examples. Distinguishing between different forms of motion and other phenomena can be achieved.
- Image-based and short-term motion-based information are used simultaneously.
- Motion detection uses a small set of simple rectangular filters that work at any scale; motion is detected by detecting temporal differences in corresponding image blocks. The small set of filters is selected from a large set of filters using the AdaBoost approach.

- **Video tracking**

- Background modeling
 - * Tracking in video is often based on some sort of subtraction of the current frame from a model of the background scene.
 - * Naive approaches are subject to noise and subtle background evolution. Median filtering is the strongest such simple technique.
 - * Much more robust approaches exist: prominent among them is to model each pixel as mixture of Gaussians. Parameter update is done in real time with heuristic approximation.
- Kernel-based tracking
 - * Gradient-based object localization and tracking can be accomplished using a general correlation criterion.
 - * Kernel-based tracking is highly efficient and facilitates tracking in real time.
 - * It is based on spatially masking the target object with an isotropic kernel, followed by applying a smooth similarity function, which reduces the tracking problem to a maximum similarity search in the vicinity of the previous location.
 - * The similarity optimization is performed using the mean shift algorithm.
- Object path analysis
 - * If several independently moving objects are tracked, the solution methods often rely on motion constraints and minimize a **path coherence function** that represents a measure of agreement between the derived object trajectory and the motion constraints.

- **Motion models**

- A predictor-corrector mechanism may be used to estimate object movement through observed noise, and then correct the prediction.
- **Kalman filtering**
 - * Kalman filtering is an approach that is frequently used in dynamic estimation and represents a powerful tool when used for motion analysis.
 - * Kalman filtering requires the system to be linear, with observations of it to be linear functions of the underlying state. Noise, both in the system and in measurement, is assumed to be white and Gaussian.
 - * While the assumptions are often unrealistic if applied to image sequences, they represent a convenient choice.
- **Particle filtering**
 - * The Kalman filter is widely implemented but has limiting assumptions. Particle filters overcome many of these.
 - * Particle filter are based on a statistical sampling approach at each time step; the sample is adjusted on the basis of image observation.
 - * The popular realization of a particle filter in vision is CONDENSATION.
- **Track-learn-detect—TLD**
 - * Most trackers are limited by pose changes, occlusion etc.
 - * Many trackers will include a *detector* to (re-)discover objects on appearance in an image.
 - * TLD tracks and detects simultaneously, and updates its learning of an object model dynamically too.
 - * It operates at a range of scales, and usually builds on the Lucas-Kanade point tracker.
 - * While tracking, the model absorbs new appearances, and prunes out those that are regarded as improbable.

16.8 Exercises

Short-answer questions

- S16.1 Describe the differences between motion detection and moving-object detection.
- S16.2 Name the object motion assumptions, and explain their rationale.
- S16.3 Explain how cumulative difference images can be used in motion analysis.
- S16.4 What is the aperture problem? What are its consequences? How can this problem be overcome?
- S16.5 What are the two basic assumptions used for optical flow computation. Are the assumptions realistic? What problems arise if they are violated?
- S16.6 Describe two approaches that increase optical flow computation robustness when the optical flow assumptions are not valid.
- S16.7 Specify how optical flow can be used to determine:

- (a) Mutual velocity of an observer and an object
 - (b) The focus of expansion
 - (c) Distance of a moving object from the observer
 - (d) Possible collision of the object with an observer and time to collision
- S16.8 Explain the image frame rate requirements for motion analysis using optical flow and motion analysis based on correspondence of interest points.
- S16.9 Explain the concept of motion analysis based on correspondence of interest points. Why is the correspondence problem difficult?
- S16.10 Determine properties of a path coherence function and explain how it can be used for object tracking.
- S16.11 Explain how occlusion may be handled in object tracking. Why are phantom points needed?

Problems

Using a static camera, create image sequences consisting of ten image frames depicting one, three, and five moving objects on a non-homogeneous background (the sequence used in Figure 16.2 illustrates an example). These image sequence(s) will be used in the problems below.

- P16.1 Develop a program for motion detection in the sequences you have created. If the approach is based on differential image analysis, pay special attention to automated threshold determination.
- P16.2 Develop a program for motion analysis using cumulative difference images. Determine the trajectory of the moving objects.
- P16.3 Develop a program for motion analysis using moving edges, and apply it to your image sequences.
- P16.4 At time $t_0 = 0$, a point object is located at real-world coordinates $x_0, y_0, z_0 = (30, 60, 10)$ and is moving with a constant speed $(u, v, w) = (-5, -10, -1)$ towards the observer. Assuming unit focal distance of the optical system:
- (a) Determine the location of the object at the image co-ordinates (x', y') at time t_0 .
 - (b) Determine the image co-ordinates of the focus of expansion.
 - (c) Determine the time of the object collision with the observer.
- P16.5 Develop and test a function for detection of interest points.
- P16.6 Develop a program for motion analysis using correspondence of interest points. Generate frame-to-frame velocity fields, not necessarily using the subsequent frames. Apply it to image sequences with one, three, and five moving objects and assess the results.
- P16.7 Develop an image tracking program using path coherence, and use it to generate object motion trajectories.
- P16.8 Implement the simple 1D example described after Algorithm 16.8 and run it with various parameter selections to observe Condensation in its simplest case.
- P16.9 Download the TLD software from <http://info.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html> and run it on a video selection of your own: discover how distorted or occluded the model may become before being lost.

- P16.10 Make yourself familiar with solved problems and **Matlab** implementations of selected algorithms provided in the corresponding chapter of the **Matlab Companion** to this text [Svoboda et al., 2008]. The **Matlab Companion** homepage <http://visionbook.felk.cvut.cz> offers images used in the problems, and well-commented **Matlab** code is provided for educational purposes.
- P16.11 Use the **Matlab Companion** [Svoboda et al., 2008] to develop solutions to additional exercises and practical problems provided there. Implement your solutions using **Matlab** or other suitable programming languages.

16.9 References

- Aggarwal J. K. and Martin W. *Motion Understanding*. Kluwer, Boston, 1988.
- Akbarzadeh A., Frahm J. M., Mordohai P., Clipp B., Engels C., Gallup D., Merrell P., Phelps M., Sinha S., Talton B., Wang L., Yang Q., Stewenius H., Yang R., Welch G., Towles H., Nister D., and Pollefeys M. Towards urban 3D reconstruction from video. In *Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, June 2006.
- Albus J. S. and Hong T. H. Motion, depth, and image flow. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, OH, pages 1161–1170, Los Alamitos, CA, 1990. IEEE.
- Baker S. and Matthews I. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56:221–255, 2004.
- Bar-Shalom Y. and Fortmann T. *Tracking and Data Association*. Academic Press, New York NY, 1988.
- Barnard S. T. and Thompson W. B. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, 1980.
- Barron J. L., Fleet D. J., and Beauchemin S. S. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994.
- Basclé B. and Deriche R. Region tracking through image sequences. In *Proc. 5th Int. Conf. on Computer Vision*, pages 302–307, Cambridge, MA, 1995.
- Baumberg A. M. *Learning deformable models for tracking human motion*. Ph.D. thesis, School of Computer Studies, University of Leeds, Leeds, UK, 1995.
- Baumberg A. M. and Hogg D. C. An efficient method of contour tracking using active shape models. In *Proceedings of the IEEE Workshop on Motion on Non-rigid and Articulated Objects*, pages 194–199, Texas, 1994a.
- Baumberg A. M. and Hogg D. C. Learning flexible models from image sequences. In Eklundh J. O., editor, *3rd European Conference on Computer Vision*, Stockholm, Sweden, pages 299–308, Berlin, 1994b. Springer Verlag.
- Beutel J., Kundel H. L., and Metter R. L. V. *Handbook of Medical Imaging, Volume 1. Physics and Psychophysics*. SPIE, Bellingham, WA, 2000.
- Bigun J., Granlund G. H., and Wiklund J. Multidimensional orientation estimation with applications to texture analysis and optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:775–790, 1991.
- Black M. J. and Jepson A. D. Recognizing temporal trajectories using the condensation algorithm. In Yachida M., editor, *Proceedings of the 3rd International Conference on Face and Gesture Recognition*, Nara, Japan, pages 16–21. IEEE Computer Society, 1998.

- Bober M. and Kittler J. Estimation of complex multimodal motion: An approach based on robust statistics and Hough transform. *Image and Vision Computing*, 12:661–668, 1994.
- Briggs W. L., Henson V. E., and McCormick S. F. *A Multigrid Tutorial*. SIAM, Philadelphia, PA, 2nd edition, 2000.
- Bruhn A., Weickert J., Feddern C., Kohlberger T., and Schnoerr C. Combining advantages of local and global optic flow methods. In Gool L. v., editor, *Pattern Recognition - LNCS Vol. 2449*, pages 454–462, Berlin, 2002. Springer.
- Bruhn A., Weickert J., Feddern C., Kohlberger T., and Schnoerr C. Variational optic flow computation in real-time. *IEEE Transactions on Image Processing*, 14:608–615, 2005.
- Cedras C. and Shah M. Motion-based recognition: A survey. *Image and Vision Computing*, 13: 129–154, 1995.
- Comaniciu D., Ramesh V., and Meer P. Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. II, pages 142–149, Hilton Head Island, SC, 2000.
- Comaniciu D., Ramesh V., and Meer P. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–575, 2003.
- Cummins D. D. and Cummins R. *Minds, Brains and Computers - The Foundations of Cognitive Science: An Anthology*. Blackwell Publishing, 2000.
- Cutler R. and Davis L. Robust real-time periodic motion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:781–796, 2000.
- Dee H. M., Hogg D. C., and Cohn A. G. Building semantic scene models from unconstrained video. *Computer Vision and Image Understanding*, 116(3):446–456, 2012.
- Djouadi A., Snorrason O., and Garber F. The quality of training-sample estimates of the Bhattacharyya coefficient. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:92–97, 1990.
- Enkelmann W. Obstacle detection by evaluation of optical flow fields from image sequences. *Image and Vision Computing*, 9(3):160–168, 1991.
- Eustice R., Singh H., Leonard J., Walter M., and Ballard R. Visually navigating the RMS Titanic with SLAM information filters. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.
- Fisher D. J., Ehrhardt J. C., and Collins S. M. Automated detection of noninvasive magnetic resonance markers. In *Computers in Cardiology*, Chicago, IL, pages 493–496, Los Alamitos, CA, 1991. IEEE.
- Fleet D. J. and Jepson A. D. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5:77–105, 1990.
- Freeman W. T. and Adelson E. H. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.
- French A. P. *Visual Tracking: from an individual to groups of animals*. Ph.D. thesis, University of Nottingham, 2006.
- Fua P. and Leclerc Y. G. Registration without correspondences. In *CVPR '94: Computer Society Conference on Computer Vision and Pattern Recognition*, Seattle, WA, pages 121–128, Los Alamitos, CA, 1994. IEEE.
- Gauss K. F. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. F Perthes and I H Besser, Hamburg, Germany, 1809.
- Gelb A., editor. *Applied Optimal Estimation*. MIT Press, Cambridge, MA, 1974.

- Gescheider G. A. *Psychophysics: The Fundamentals*. LEA, 3rd edition, 1997.
- Gibet S., Courty N., and Kamp J., editors. *Gesture in Human-Computer Interaction and Simulation, 6th International Gesture Workshop, GW 2005, Berder Island, France, May 18-20, 2005, Revised Selected Papers*, volume 3881 of *LNCS*, 2006. Springer.
- Glazer F. Multilevel relaxation in low level computer vision. In A. R., editor, *Multiresolution Image Processing and Analysis*, pages 312–330. Springer Verlag, Berlin, 1984.
- Gu H., Shirai Y., and Asada M. MDL-based segmentation and motion modeling in a long image sequence of scene with multiple independently moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:58–64, 1996.
- Hager G. and Belhumeur P. Real-time tracking of image regions with changes in geometry and illumination. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 403–410, San Francisco, CA, 1996.
- Horn B. K. P. *Robot Vision*. MIT Press, Cambridge, MA, 1986.
- Horn B. K. P. and Schunk B. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.
- Hu X. and Ahuja N. Motion and structure estimation using long sequence motion models. *Image and Vision Computing*, 11:549–569, 1993.
- Huang C. L. and Chen Y. T. Motion estimation method using a 3D steerable filter. *Image and Vision Computing*, 13:21–32, 1995.
- Hummel R. and Sundareshwaran. Motion parameter estimation from global flow field data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:459–476, 1993.
- Intille S. and Bobick A. Visual tracking using closed-worlds. In *5th International Conference on Computer Vision*, Boston, USA, pages 672–678, 1995.
- Isard M. and Blake A. Condensation—conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, 1998.
- Jain R. Dynamic scene analysis using pixel-based processes. *Computer*, 14(8):12–18, 1981.
- Jain R. Direct computation of the focus of expansion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):58–64, 1983.
- Jain R. Difference and accumulative difference pictures in dynamic scene analysis. *Image and Vision Computing*, 2(2):99–108, 1984.
- Jain R., Martin W. N., and Aggarwal J. K. Segmentation through the detection of changes due to motion. *Computer Graphics and Image Processing*, 11:13–34, 1979.
- Jain R., Kasturi R., and Schunck B. G. *Machine Vision*. McGraw-Hill, New York, 1995.
- Julier S. and Uhlmann J. A new extension of the Kalman filter to nonlinear systems. In *Proceedings SPIE, Vol. 3068*, pages 182–193, Bellingham, WA, 1997. SPIE.
- Kaernbach C., Schroger E., and Muller H. *Psychophysics Beyond Sensation: Laws and Invariants of Human Cognition*. LEA, 2003.
- Kalal Z., Matas J., and Mikolajczyk K. Online learning of robust object detectors during unstable tracking. In *Proceedings of the IEEE On-line Learning for Computer Vision Workshop*, pages 1417–1424, 2009.
- Kalal Z., Matas J., and Mikolajczyk K. P-n learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, pages 49–56, 2010.
- Kalal Z., Mikolajczyk K., and Matas J. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, 2012.
- Kalman R. E. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, 82:35–45, 1960.

- Kearney J. K. and Thompson W. B. Bounding constraint propagation for optical flow estimation. In Aggarwal J. K. and Martin W., editors, *Motion Understanding*. Kluwer, Boston, 1988.
- Kearney J. K., Thompson W. B., and Boley D. L. Optical flow estimation—an error analysis of gradient based methods with local optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):229–244, 1987.
- Khan Z., Balch T., and Dellaert F. Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):1805 – 1918, 2005.
- Koenderink J. J. Optic flow. *Vision Research*, 26(1):161–180, 1986.
- Lucas B. D. and Kanade T. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI’81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. URL <http://dl.acm.org/citation.cfm?id=1623264.1623280>.
- MacCormick J. and Blake A. A probabilistic exclusion principle for tracking multiple objects. In *International Conference on Computer Vision*, Corfu, Greece, pages 572–578, 1999.
- Magee D. R. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 22(2):143–155, 2004.
- Magee D. R. and Boyle R. D. Detecting Lameness using ‘Re-sampling condensation’ and ‘Multi-stream Cyclic Hidden Markov Models’. *Image and Vision Computing*, 20(8):581–594, 2002.
- McFarlane N. J. B. and Schofield C. P. Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8:187–193, 1995.
- Nagel H. H. On the estimation of optical flow: Relations between different approaches and some new results. *Artificial Intelligence*, 33:299–324, 1987.
- Needham C. J. and Boyle R. D. Tracking multiple sports players through occlusion, congestion and scale. In *Proc. British Machine Vision Conf.*, pages 93–102, 2001.
- Negahdaripour S. and Ganesan V. Simple direct computation of the FOE with confidence measures. In *Proceedings, 1992 Computer Vision and Pattern Recognition*, Champaign, IL, pages 228–233, Los Alamitos, CA, 1992. IEEE.
- Oliver N., Rosario B., and Pentland A. A Bayesian computer vision system for modeling human interactions. In Christensen H. I., editor, *Proceedings of ICVS99*, Gran Canaria, Spain, pages 255–272. Springer Verlag, 1999.
- Power P. W. and Schoonees J. A. Understanding background mixture models for foreground segmentation. In Kenwright D., editor, *Proceedings, Imaging and Vision Computing New Zealand*, Auckland, NZ, 2002.
- Puzicha J., Rubner Y., Tomasi C., and Buhmann J. Empirical evaluation of dissimilarity measures for color and texture. In *Proc. 7th Int. Conf. on Computer Vision*, pages 1165–1173, Kerkyra, Greece, 1999.
- Rangarajan K. and Shah M. Establishing motion correspondence. *CVGIP – Image Understanding*, 54:56–73, 1991.
- Ringach D. L. and Baram Y. A diffusion mechanism for obstacle detection from size-change information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:76–80, 1994.
- Scott G. L. *Local and Global Interpretation of Moving Images*. Pitman–Morgan Kaufmann, London–San Mateo, CA, 1988.

- Sethi I. K. and Jain R. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, 1987.
- Shapiro V., Backalov I., and Kavardjikov V. Motion analysis via interframe point correspondence establishment. *Image and Vision Computing*, 13:111–118, 1995.
- Shi J. and Tomasi C. Good features to track. *Computer Vision and Pattern Recognition*, pages 593–600, 1994.
- Sorenson H. W. Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum*, pages 7–12, 1970.
- Stauffer C. and Grimson W. E. L. Adaptive background mixture models for real-time tracking. In *CVPR '99: Computer Society Conference on Computer Vision and Pattern Recognition*, Ft. Collins, USA, volume 2, pages 246–252, 1999.
- Subbarao M. *Interpretation of Visual Motion: A Computational Study*. Pitman–Morgan Kaufmann, London–San Mateo, CA, 1988.
- Subbarao M. Bounds on time-to-collision and rotational component from first-order derivatives of image flow. *Computer Vision, Graphics, and Image Processing*, 50(3):329–341, 1990.
- Svoboda T., Kybic J., and Hlavac V. *Image Processing, Analysis, and Machine Vision: A MATLAB Companion*. Thomson Engineering, 2008.
- Thompson W. B. and Barnard S. T. Lower level estimation and interpretation of visual motion. *Computer*, 14(8):20–28, 1981.
- Thompson W. B., Mutch K. M., and Berzins V. A. Dynamic occlusion analysis in optical flow fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):374–383, 1985.
- Thompson W. B., Lechleider P., and Stuck E. R. Detecting moving objects using the rigidity constraint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:162–166, 1993.
- Trottenberg U., Oosterlee C., and Schueller A. *Multigrid*. Academic Press, Dan Diego, CA, 2001.
- Tsao A. T., Hung T. P., Fuh C. S., and Chen Y. S. Ego-motion estimation using optical flow fields observed from multiple cameras. In *Computer Vision and Pattern Recognition*, pages 457–462, Los Alamitos, CA, 1997. IEEE Computer Society.
- Ullman S. *The Interpretation of Visual Motion*. MIT Press, Cambridge, MA, 1979.
- Vega-Riveros J. F. and Jabbour K. Review of motion analysis techniques. *IEE Proceedings, Part I: Communications, Speech and Vision*, 136(6):397–404, 1989.
- Viola P. and Jones M. Rapid object detection using a boosted cascade of simple features. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 511–518, Kauai, Hawaii, 2001. IEEE.
- Viola P., Jones M., and Snow D. Detecting pedestrians using patterns of motion and appearance. In *Proc. Int. Conf. Computer Vision*, pages 734–741, Nice, France, 2003.
- Watson A. B. and Ahumada A. J. Model of human-model sensing. *Journal of the Optical Society of America*, 2:322–342, 1985.
- Weickert J. and Schnoerr C. A theoretical framework for convex regularizers in PDE-based computation of image motion. *International Journal of Computer Vision*, 45:245–264, 2001.
- Wu S. F. and Kittler J. General motion estimation and segmentation. In *Visual Communications and Image Processing '90*, Lausanne, Switzerland, pages 1198–1209, Bellingham, WA, 1990. SPIE.
- Young D. M. *Iterative Solution of Large Scale Linear Systems*. Academic Press, New York, NY, 1971.

Index

Symbols

2.5D sketch, 586–587, 655
2D co-ordinate system, 646
2D projection, 331, 332, 657, 670, 788
2D shape, 101, 329, 331
3D information, 329, 645
3D interpretation, 585, 646, 657, 658
3D model, 655
3D object, 655
3D representation, 13, 586, 656
3D shape, 101, 329, 332

A

A-algorithm, 197–204, 209
AAM, 492–502
aberrations, 86
accuracy, 241
ACRONYM, 474, 662
active appearance model, *see* AAM
active perception, 587
active sensor, 628
active shape model, *see* ASM
active vision, 648
acuity, 25
AdaBoost, 440–442, 513, 514, 516, 811–813
additive noise, 28
adjacency, 281
affinity, *see* fuzzy affinity
AGC, *see* automatic gain control
albedo, 94
aliasing, 63
anti-aliasing, 65
anti-extensive transformation, 691
aperture problem, 156, 793, 806
aperture stop, 84
arc (of a graph), 106

area, 103, 110, 337, 354–355, 358, 662, 684
area-based stereo, 624
ASM, 487–491
aspect, 670
aspect graph, 671
autocorrelation, 728
automatic gain control, 42
Autostitch, 479, 481

B

B-reps, 662
back-tracking, 208, 418, 424, 473, 530
background, 18
 uneven, 698
background modeling, 815, 820
ball, 700
 geodesic, 708
 maximal, 700
 unit, 700
balloon, 267
band-limited, 63
baseline, 610, 615
bay, 23
Bayes formula, 395, 504, 506–508
Bayesian belief networks, 551–552
bin-picking, 625
blackboard, 473
blocks world, 656
blooming, 41
blur
 atmospheric turbulence, 163
 Gaussian, 24, 140, 145
 motion, 163
 smoothing, 125
 wrong focus, 163
body reflection, 31

854 Index

- boosting, 439–442, 513–517, 810–814
 - border, 21
 - detection, **187**, 431
 - optimal, **196–210**, 288, **295–303**
 - simultaneous, 289
 - extended, 193–195
 - inner, 21, 191, 192
 - inter-pixel, 193
 - occlusion, 632
 - outer, 21, 191–194
 - boundary, *see* border
 - BRDF, 93
 - brightness, **12**, 13, 90, 101, 118, 128, 358, 359, 506, 631, 750
 - correction, 117
 - interpolation, 123–124
 - transformation, 117–120
 - brightness interpolation
 - bi-cubic, 124
 - linear, 124
 - nearest neighbor, 123
 - C**
 - C4.5, 442
 - CAD, 502
 - calculus, 133, 427, 684
 - calibration matrix, 600
 - camera, 11, 42, 101, 350, 353, 787, 788, 794
 - analog, 42
 - jitter, 42
 - Canny edge detector, 136, 144–147
 - canonical configuration, 615
 - CCD, 41
 - cellular complex, 19
 - center of gravity, 358, 371, 372
 - central limit theorem, 126
 - CFL condition, 280
 - chain, 104
 - code, 104, 195, **336–339**, 344, 354
 - Markov, 769
 - chamfering, 19, 235
 - Chan-Vese functional, 279
 - characteristic polynomial, 73
 - characteristic strip, 631
 - characteristic view, 670
 - class, 390
 - equivalence, 18
 - identifier, 392
 - classification
 - contextual, **505–509**, 530
 - recursive, 507
 - classifier, 390, 417
 - base, 439
 - boosted
 - cascade, 513–517, 810–814
 - learning, 393, 396, **398–400**, 420
 - linear, 392–393
 - maximum likelihood, 395
 - minimum distance, 393, 407
 - minimum error, 395–400
 - nearest neighbor, 393
 - NN, 393
 - non-linear, 392
 - random forest, 442–448
 - bagging, 446
 - C4.5, 442
 - classification, 442, 446
 - decision making, 446
 - ensemble model, 448
 - information gain, 445
 - manifold forest, 448
 - manifold learning, 448
 - predictor, 443
 - regression, 442, 446
 - semi-supervised forest, 448
 - semi-supervised learning, 448
 - stopping criterion, 446
 - supervised learning, 448
 - training, 444
 - unsupervised learning, 448
 - setting, 395, 397
 - support vector machine, 400, 511
 - syntactic, 417
 - weak, 439, 513
- clique, 424, 425
 - closing, 692–693
 - cluster analysis, 38, 406, 407
 - fuzzy
 - k-means, 407
 - mean shift, 255, 261
 - CMOS, 41
 - CMY, 37
 - co-occurrence matrix, 102
 - co-ordinate system

- object-based, 13, 655
- polar, 336
- rectangular, 335
- tangential, 336, 337
- COBIUS, 473
- code
 - dictionary, 730, 734
 - Freeman, 104
 - leaf, 111
 - run length, 105
- coding
 - Huffman, 733, 734
 - low and high frequencies, 733
 - region border, 733
- collineation, 592
- color
 - constancy, 39
 - gamut, 34
 - hue, 37
 - image, 30
 - L,u,v, 259
 - map, 38
 - metamer, 32
 - model, 33
 - palette, 734
 - perceived, 259
 - primary, 30
 - saturation, 37
 - secondary, 36
 - table, 38
 - value, 37
 - YIQ, 36
 - YUV, 37
- combination
 - convex, 34
 - linear, 51
- compatibility function, **525**, 536, 537
- compression, 5, 7, 236, 722–740
 - application
 - asymmetric, 730, 739
 - symmetric, 730, 739
 - dictionary-based, 734
 - DPCM, **728**, 730
 - fractal, 733
 - hierarchical, 730–732
 - hybrid, 722, 730, 731, **732**
 - JPEG, 734–736
 - JPEG-2000, 736–738
 - Lempel-Ziv, 734
 - MJPEG, 736, 738, 739
 - MJPEG-2000, 738
 - motion JPEG, *see* MJPEG, compression
 - MPEG, 734, 738–740
 - predictive, 722, **727–728**, 732
 - progressive, 730–732
 - pyramid, 730
 - ratio, **724**, 727, 728, 731
 - region growing, 733
 - scalability
 - quality, 736
 - resolution, 736
 - spatial, 736
 - smart, 731, 732
 - transform, 722, **724**, 732
 - vector quantization, **730**, 732
 - wavelet, 727
- computer aided diagnosis, 502
- computer graphics, 13, 101, 346, 772
- Condensation, 831, **837–840**
- cone, 32
- confidence, 507–539
- configuration
 - canonical, 615
 - degenerate, 595
 - rectified, 615
- conic, 351, 352
- connected component labeling, 333
- connectedness, *see* fuzzy connectedness
- connectivity
 - eight, 19
 - four, 18
 - fuzzy, 280–288
 - problems, 19
- constraint
 - propagation, **524–528**, 531, 532, 657
- constructive solid geometry, 660
- context, 178, 465, 469, **505–524**, 531, 532
- contiguous
 - multiple, 18
 - simple, 18
- contour
 - false, 15
 - partitioning, 344

- contrast, **25**, 118, 157, 188, 232, 753, 754, 791
 - control strategy, 386, **465–474**
 - bottom-up, 205, 385, **467–468**, 469, 471, 527
 - combined, 469
 - hierarchical, 466, 472
 - model-based, 468, 470
 - non-hierarchical, 472
 - parallel, 466, 468
 - serial, 466
 - top-down, 467–469, 471
 - convex hull, 22, 684
 - convex region, 22
 - convolution, **51**, 123, 148, 149
 - discrete, 52
 - mask, 52, 125
 - theorem, 59
 - coordinate system
 - object-based, 587
 - corner, 156
 - definition, 156
 - detector, 147, 155, 156
 - Harris, 157, 805
 - Moravec, 157, 805
 - correction, gamma, 42
 - correlation, 80, 348, 722–724, 727
 - projective, 611
 - correspondence, 121, 156, 621, 645, 646, 788–790, 804, 807–809, 829
 - correlation-based, 624
 - feature-based, 625
 - landmark, 490, 497
 - point, 490, 497
 - problem, 156, 626, 645, 646
 - time, 498
 - cost function, 197–204, 311
 - automated design, 210
 - edge based, 311–312
 - Gibbs model, 296
 - region based, 279, **312–313**
 - cost transform, 202, 210
 - covariance matrix, 75
 - crack edge, 21
 - criterion
 - detection, 144
 - localization, 144
 - one response, 144
 - cross ratio, 350
 - crossover, 428–430
 - curvature, 337
 - deformation, 274
 - peak, 345
 - primal sketch, 346
 - curve
 - decomposition, 346
 - detection, 212, 214–216, 218
 - digital straight segment, 337, 343
 - DSS, 337
 - granulometric, 711
 - cyclopean image, 626
 - cyclopean separation, 626
- ## D
- data structure, 100
 - hierarchical, 108
 - relational, 107
 - traditional, 101
 - de-fuzzification, 435, 438
 - composite maximum, 438
 - composite moments, 438
 - decimation, 664
 - decision rule, 392, 398
 - deficit of convexity, 23
 - deformable objects, 655
 - deformation
 - constant, 274
 - curvature, 274
 - degenerate configuration, 595
 - degradation, 163
 - atmospheric turbulence, 163
 - relative motion, 163
 - wrong lens focus, 163
 - depth, 13, 651
 - of field, 85
 - of focus, 84
 - depth map, 586, 628, 650
 - diagram, Voronoi, 531
 - Dice evaluation, 237
 - difference of Gaussians, *see* DoG
 - diffusion, generalized, 269
 - dilation, 688–689
 - conditional, 714
 - geodesic, 708
 - gray-scale, 694–697
 - Dirac distribution, 51

- discrete convolution, 52
- discrimination function, 392, 396
- disparity, 555, 616, 626
 - gradient, **625**, 626
 - gradient limit, 625
- distance, 17
 - chessboard (D_8), 17
 - city block (D_4), 17
 - Euclidean (D_E), 17
 - function, 19, 707, 717
 - geodesic, 707
 - Hausdorff, 238
 - Levenshtein, 425
 - Mahalanobis, 407
 - Sampson, 597
 - transform, 19
- distribution function, 80
- DoG, 141, 474
- domain
 - range, 260
 - spatial, 260
- DSS, 343
- duality, 591
- duality, morphological, 686
- dyadic, 69
- dynamic programming, 206–210
 - intelligent scissors, 209, 268
 - live lane, 210
 - live wire, 209, 268
- E**
- Ebbinghaus illusion, 26
- edge, 5, 7, **21**, 586
 - chain, 187, 205
 - crack, 21, 191, 193, 222, 223
 - detector, 431
 - direction, 134
 - magnitude, 134
 - relaxation, **190–191**, 205
- edge detector, 133–143, 585
 - Canny, 144–147, 755
 - facet model, 147
 - in multi-spectral image, 148
 - Kirsch, 139
 - Laplace, 135, 137
 - Marr-Hildreth, 139
 - parametric, 136, 147
 - Prewitt, 137
 - Roberts, 137
 - Sobel, 138
 - zero-crossing, 139
- effect, blooming, 41
- eigen-image, 76
- eigenvalues
 - multiple, 73
- elastics, 468
- EM, *see* expectation-maximization
- entropy, **24**, 722–724, 754
 - condition, 279
- epipolar
 - constraint, 606, 610, 611, **622**
 - line, 610, 627
 - plane, 610
- epipolar constraint, 627
- epipole, 610
- equation
 - irradiance, 95
- erosion, 689–692
 - geodesic, 708
 - gray-scale, 694–697
 - ultimate, 705
- essential matrix, 612
- estimation
 - density, 255, 262
 - gradient, 257
 - mode, 258
 - non-parametric, 257
- estimator
 - multivariate density, 256
- Euler-Poincaré characteristic, 355
- event (aspect), 671
- exemplar, 393, 406, 412, 413
- expansion
 - isotropic, 688
- expectation-maximization, 183, 548, **556–564**, 817
- F**
- face detection, 442, 513, 514, 516
- facet, 147
- fast marching, 278
- feature, 386
 - cascade
 - boosted, 513
 - discriminativity, 397
 - informativity, 397

- simple, 513
 - space, 255, 391
 - color, 259
 - synthesis, 146
 - vector, 386
- feedback, 6, 188, 732
- fill, morphological, 688
- filter, 116–165
 - bank, 765–767
 - Gaussian, 586
 - median, 130–132, 816
 - particle, *see* particle filter
 - separable, 127
- filtering
 - averaging, 125
 - band-pass, 149, 150
 - discontinuity preserving, 255
 - discontinuity-preserving, 261
 - high-pass, 149, 150
 - inverse, 163
 - Kalman, 162
 - low-pass, 149
 - mean shift, 255, 261
 - median, 130
 - Wiener, 164
- fitness function, 535
- focal point, 83, 598
- Forward algorithm, 545
- fractal, 348, 770
 - dimension, 337, 762
- frame, 390
- free-form surface, 655
- Freeman code, 104
- Fresnel lens, 85
- Frobenius norm, 615
- front propagation, 273
- function
 - autocorrelation, 80
 - autocovariance, 80
 - continuous, 11
 - cross correlation, 80, 841
 - cross covariance, 80
 - digital, 11
 - Dirac, 51
 - discrete, 11
 - distance, 19, 707
 - distribution, 80
 - fitness, 535
 - kernel, 403
 - level set, 276
 - partition, 553
 - point spread, 53
 - quench (morphology), 704
 - speed, 274
- functional
 - Chan-Vese, 279
 - Mumford-Shah, 279
- fundamental matrix, 610
- fuzzy
 - adjacency, 281
 - affinity, 281
 - complement, 434
 - composition, 435
 - min-max, 436
 - connectedness, 282
 - absolute, 288
 - hard-adjacency, 281
 - map, 282
 - connectivity, 280–288
 - 2-object, 286
 - absolute, 282, 286
 - iterative, 286
 - multiple-object, 286
 - relative, 286
 - scale-based, 286
 - correlation
 - minimum, 436
 - product, 437
 - intersection, 434
 - logic, 281, 388, 432–439
 - membership function
 - maximum normal form, 433
 - minimum normal form, 433
 - reasoning
 - monotonic, 435
 - set, 432
 - hedge, 433
 - space, 432
 - system, 432–439
 - model, 435
 - union, 434

G

- gamma correction, 42
- gamut, 34
- ganglion cell, 143

- Gaussian filter, 140, 141, 586
 - Gaussian mixture models, 407, **556–564**, 762, 817–820
 - Gaussian noise, 28
 - generalized cylinders, 661
 - genetic algorithm, 427–430, 439, 535–542
 - genus, 355
 - geodesic transformation, 707
 - Geographical Information Systems, 111
 - geometric signals, 628
 - geometric transformation, 120–124
 - Gestaltist theory, 645
 - GIF, 734
 - GIS, 111
 - Golay alphabet, 702
 - gradient descent, 409, 494, 500, 548, 634
 - gradient operator, 125, 133–143
 - approximated by differences, 135
 - Kirsch, 139
 - Laplace, 137
 - Prewitt, 137
 - Roberts, 137
 - Sobel, 138
 - gradient space, 92
 - gradient vector flow, *see* snake
 - grammar, **387**, 414–420, 770–773
 - context-free, 416
 - context-sensitive, 416
 - fuzzy, 417
 - general, 416
 - inference, 414, **420–421**
 - non-deterministic, 417
 - regular, 417
 - stochastic, 417
 - granulometry (morphological), 711
 - graph, **106**, 193, 197, 206, 235, 353, 365, 387, 414, 418, 421–426
 - arc, 106
 - assignment, 425
 - constraint
 - hard, 296, 300, 303, 311, 317
 - soft, 296, 311, 317
 - surface separation, 309
 - construction, 306
 - geometric, 303
 - graph cuts, 295–303
 - seeds, 296
 - isomorphism, 421–426
 - labeled, 389
 - maximum flow, 295–303
 - augmenting path, 298
 - push-relabel, 298
 - residual graph, 298
 - minimum closed set, 310
 - minimum s-t cut, **295–317**
 - neighborhood, 372
 - node, 106
 - region, 365, 370
 - region adjacency, 106, 112, 179, 223, 225, 372, 528, 531, 532, 535–538
 - search, **196–203**, 209, **295–317**, 469, 470
 - advanced approaches, 288
 - heuristic, **199–205**, 209
 - LOGISMOS, 303–317
 - three-dimensional, 289
 - similarity, 421, 425, 426
 - weighted, 106, 193, 197, 421, 422
 - graph cut, 554
 - gray-scale transformation, 117
 - greedy exchange algorithm, 828
 - grid, 14
 - hexagonal, 15
 - square, 15
 - groundplane, 675
 - group, 349, 352
 - Lie, 349
 - plane-projective, 349
 - grow, morphological, 688
 - GVF, *see* snake
- ## H
- hanging togetherness, 281
 - Hausdorff distance, *see* distance, Hausdorff
 - HEARSAY, 474
 - heuristic, 199, 201, 204, 209, 222, 232, 531, 532, 542
 - hidden Markov model, **543–551**, 561, 831
 - Baum-Welch algorithm, 548, 549, 563, 564
 - coupled, 549–551
 - decoding, 544, 546–548

- evaluation, 544–545
- forward algorithm, 545
- Forward-Backward algorithm, 548, 563, 564
- learning, 544, 548
- second order, 549
- Viterbi algorithm, 546, 548
- hidden Markov tree, 768
- histogram, **23**, 179, 181–183, 186, 221, 723, 724
 - bimodal, 182
 - cumulative, 119
 - equalization, 37, 118–120
 - multi-modal, 182
 - smoothed, 186
 - transformation, 183
- histograms of oriented gradients, *see* HOG
- hit-or-miss transformation, 692
- HOG, 509
- hole, 18
- homogeneity, 220, 221, 224–232
- homogeneous coordinates, 591
- homography, 481, 592
- homotopic substitute (of skeleton), 702, 703
- homotopic transformation, 699
- horizon, 652, 675
- Hough transform, 210–217
- HSV, 37
- hue, 37
- human visual system, 586
- hypothesis, 468, 535–542
- hypothesize and verify, 345, 373, **468**, 470, 535
- hysteresis, 146, 188
- I**
- ICP algorithm, 667
- illumination, 90
- image
 - binary, 102
 - co-ordinates, 13
 - color, 30
 - compression, *see* compression
 - cyclopean, 626
 - databases, 236
 - difference, 790–793
 - digitization, 14–16
 - dynamic, 14
 - element, 15, 281
 - function, 11
 - iconic, 101
 - indexed, 38
 - integral, 103, 513, 514, 812
 - interpretation, 465, 542
 - irradiance equation, 95
 - multispectral, 32
 - palette, 38
 - plane, 598
 - pre-processing, 116–165
 - pseudo-color, 38
 - quality, 27
 - reconstruction, 722–728, 731
 - rectification, 616, 617
 - registration, 156, 233, 561, 664
 - representation, 5
 - scale-space, 144, 345
 - segmented, 101
 - sharpening, 5, 135
 - skew, 122
 - static, 13
 - understanding, 8, **464–542**
 - random forest, 517–524
- imaging modalities, 12
- implementation, in Marr's theory, 585
- importance sampling, 838
- impossible objects, 658
- impulse
 - Dirac, 51
 - limited, 64
- increasing transformation, 689, 691
- inference, 465
- information entropy, 24
- integral image, *see* image, integral
- intelligent scissors, 209, 268
- intensity image, 12, 13, 116, 139, 467, 633
- interest point, 155, 156, 625
- interpretation
 - genetic, 534, 535
 - tree, 530
- interval tree, 144, 345
- invariants, 332, 349, 350, 352
 - moment, 359
 - scalar, 350

- inverse filtering, 163
- inverse transformation, 692
- irradiance, 4, 42, 90
 - equation, 632
 - spectral, 93
- Ising model, 554
- ISODATA, 407, 504
- isotropic, 140, 151, 158, 690, 693, 789, 820, 821, 845
- isotropic expansion, 688
- iteration
 - Gauss-Seidel, 796, 800
 - multigrid, 801
- J**
- Jacobian determinant, 122
- jitter, 42
- Jordan block, 73
- just-enough-interaction, 313
- K**
- K-D tree, 394, 475, 730
- K-means, **406**, 556, 559, 560
- Kalman filter, 162, 831–837
 - extended, 833
 - unscented, 833
- Kalman gain matrix, 832
- kernel, 403
 - bandwidth, 257
 - Epanechnikov, 256, 257
 - normal, 256
 - polynomial
 - homogeneous, 403
 - non-homogeneous, 403
 - radial basis function, 403
 - Gaussian, 403
 - trick, 403
- Kinect, 517
- knowledge, 6, 8, **386–390**, 428, 430
 - a priori, 187, 188, 196, 211, 219, 225, 331, 470, 473, 503, 532, 536, 540
 - base, 386–388
 - procedural, 388
 - representation, 386–390, 465
- L**
- label, 333–335, 473, 503–509, 524–530, 532, 533, 536–538
 - collision, 333, 335
- labeling, 333, 354, 473, 504, 505, 507, 508, **524–542**
 - consistent, 524, 526
 - discrete, **525**, 530
 - probabilistic, 524, **527**
 - semantic, 524
- lacunarity, 763
- Lagrange multipliers, 795
- lake, 23
- Lambertian surface, 94
- landmarking, 490, 497
- landmarks, 373
- language, 414–420
- Laplacian, 135, 586
- Laplacian of Gaussian, 140
- learning, **397–406**, 410, 415, 420, 430
 - from experience, 465
 - unsupervised, 406
- least squares, 477
- leave *K* out testing, 397
- leave one out testing, 397
- length
 - focal, 84
- lens
 - Fresnel, 85
 - telecentric, 85
 - thick, 85
 - thin, 83
- level sets, 273–280
 - fast marching, 278
 - front propagation, 273
 - narrow band, 278
- LIDAR, 628
- light, 117, 143
 - source, 13, 631, 632
- line
 - finding, 155
 - labeling, 656–658
 - thinning, 155
- linear combination, 51
- linear system, 53
- linearity, 50
- linearly separable, 391
- linguistic
 - variable, 388, 434, 435
- live lane, 210
- live wire, 209, 268

local shading analysis, 634
 location of interest, 155
 LOGISMOS, 303–317
 Lucas–Kanade point tracking, 234, 800,
 805–807, 841
 luminance, 37
 luminous efficacy, 89
 luminous flux, 89

M

magnification, 84
 map
 cumulative edge, 146
 depth, 651
 edge, 486
 fuzzy connectedness, 282
 reflectance, 94, 633
 region, 106
 marker, 713
 Markov assumption, 837
 Markov chain, 769
 Markov model, 543
 Markov Random Fields, *see* MRF
 Marr (David), 8, 585
 Marr's theory, 139, 467, 585–587, 656
 Marr-Hildreth edge detector, 136, 139,
 145
 matching, 232–236, 426, 427, 465, 474–
 477
 chamfer, 235
 graphs, 421
 relational structures, 419
 subgraphs, 425
 mathematical morphology, 369
 matrix, 101
 calibration, 600
 co-occurrence, 102
 covariance, 75
 essential, 612
 fundamental, 610
 Jacobian, 494
 projection, 601
 maximal ball, 700
 MCMC, 840
 MDL, 665, 829
 mean shift, 255–262, 407
 basin of attraction, 258, 262
 medial axis, 370
 median, 130
 memory
 long-term, 473
 short-term, 473
 metric, 17, 822
 Mexican hat, 141
 Minkowski algebra, 688
 model, 178, 203, 204, 217, 220, 464, 465,
 467–473, 534
 3D, 655
 active contour, **263–273**, 789
 geodesic, 273–280
 geometric, 273–280
 level set, 273–280
 deformable, 263, 273
 geometric, 273–280
 parametric, 273
 facet, 147, 155, 157
 quadric surface, 663
 surface, 655
 volumetric, 655, 658
 Moiré interferometry, 630
 moment
 invariant, 359
 affine, 359
 statistical, 57, 110
 Moravec detector, 805
 morphological transformation, 686
 quantitative, 687
 morphology, 684–716, 768
 mother wavelet, 67
 motion, 644, 787–843
 analysis, 787–843
 correspondence of interest points,
 788, 789, **804–829**
 differential, 789–793
 assumptions, 789, 826, 829
 continuous, 648
 correspondence of interest points,
 830
 cyclic, 830
 description length, 829
 events, 830
 features, 830
 field, 788
 gesture interpretation, 830
 lipreading, 830
 path analysis, 826–830

- path coherence, 826
 - deviation, 826
 - function, 826, 827
- recognition, 830
- relative, 830
- rotational, 646, 648, 801, 803
- trajectory
 - parameterization, 830
 - translational, 646, 648, 801, 803
 - verb recognition, 830
- motion tracking, 814
- MP4, 740
- MRF, 295, 553–555, 761
- multi-view representation, 670
- multiple contiguous, 18
- Mumford-Shah functional, 279
- mutation, 428–430
- N**
 - nats, 560
 - nearest neighbor, *see* classifier, NN
 - Necker cube, 585
 - neighbor, 17
 - neural nets, 407–413, 439
 - adaptive resonance theory, 412
 - back-propagation, 409–410
 - epoch, 410
 - feed-forward nets, 409–410
 - gradient descent, 409
 - Hopfield, 407, **412–413**
 - Kohonen networks, 407, **411–412**
 - momentum, 410
 - transfer function, 408
 - unsupervised learning, 411–412
 - NN classifier, *see* classifier, NN
 - node (of a graph), 106
 - noise, 5, 7, 28, 330, 336, 339, 354, 359, 360, 369, 505
 - additive, 28
 - Gaussian, 28, 831, 832
 - impulse, 30
 - multiplicative, 30
 - quantization, 30
 - salt-and-pepper, 30
 - white, 28, 831
 - non-maximal suppression, 145, 188
 - norm
 - L2, 512
 - L2-Hys, 512
 - NTSC, 14
 - NURBS, 662
- O**
 - object, 18
 - coloring, 333
 - connected component labeling, 333
 - description, 391
 - formal, 391
 - qualitative, 391, 413
 - quantitative, 391, 413
 - relational, 413
 - identification, 333–335
 - impossible, 658
 - labeling, 333, 354
 - recognition, 385–431, 474
 - reconstruction, 329, 330
 - objective function, 529, 530, 533–537
 - occlusion, 330, 342, 373, 658, 790, 836
 - occupancy grid, 658
 - OCR, 7, 233, 329, 355, 544, 545, 548
 - octrees, 111
 - opening, 692–693
 - operator
 - morphological, 686
 - Zadeh, 434
 - optical axis, 598
 - optical center, 598
 - optical flow, 633, 648, 788, **794–804**, 809
 - computation, 788, 789
 - field, 648
 - global and local estimation, 797
 - method
 - CLG, 800
 - combined local-global, 800
 - least square, 800
 - variational, 800
 - real-time, 800
 - optimization, 206, 207, 235, 397, 400, 412, **426–431**, 487
 - orthographic
 - projection, 13, 646, 648
 - view, 646
 - OUTEX, 762
 - over-fitting, 400, 441

P

- PAL, 14
- palette, 38, 734
- parallel implementation, 8, 102, 108, 195, 217, 466, 509, 527, 529, 542, 627, 689, 797
- particle filter, 831, **837–840**
- partition function, 553
- path, 17
 - simple, 195
- pattern, 391
 - space, 391
 - vector, 391
- pattern recognition, 385–421
 - capacity, 401
 - statistical, 390, 413, 414
 - syntactic, 413–421
- PCA, *see* principal component analysis
- PDM, **481–491**, 817
 - alignment, 482
 - covariance matrix, 484, 485
 - eigen-decomposition, 484
 - landmark, **482**, 483, 485, 487–489
 - polar, 490
 - polynomial regression, 490
- perception, 25, 27, 465, 769
 - human, 15, 25, 651
 - visual, 25
- perceptron, 409
- perimeter, 193, 222, 223, 332, 337, 684
- perspective
 - projection, *see* projection, perspective, 648
- photo-emission, 40
- photometry, 89
- photosensitive, 40
- picture element, 15
- pixel, 15
 - adjacency, 17
 - hexagonal, 15
- planning, 465
- point
 - focal, 83
 - of interest, 155
 - principal, 83
 - representative, 685
 - sampling, 14
 - sets (morphological), 685
 - simple, 368
 - vanishing, 609, 675
- point distribution model, *see* PDM
- polynomial
 - characteristic, 73
- post-processing, 505, 534, 535, 542
- power spectrum, 31, 81
- pre-processing, 116–165, 467, 506
 - classification, 116
 - local, 125
- precision, 241
- predicate logic, 387, 389
- predictor-controller, 831
- primal sketch, 586–587, 655
 - curvature, 346
- primitive
 - texture, 651
 - volumetric, 587
- principal component analysis, 74, 820
- principal point, 83
- principle
 - superposition, 51
 - uncertainty, 57
- probability, 80
 - density, 80, 398
 - estimation, 398
 - distribution, 80
- problem
 - aperture, 156
- production
 - rules, 387
 - system, 387, 388, 465, 473
- projection, 77, 355, 646, 648, 684
 - histogram, 355
 - orthographic, 13, 646, 648
 - parallel, 13
 - perspective, 12, 13, 589, 648
- projection matrix, 600
- projective reconstruction, 605
- projective transformation, 592
- PROLOG, 387
- propagation theory, 552
- pseudo-color, 38, 120
- purposive vision, 589
- pyramid, **108**, 225–227, 466, 730, 731
 - equivalent window, 111
 - irregular, 112
 - Laplacian, 112

- M-pyramid, 108
 - matrix, 108
 - reduction factor, 111
 - reduction window, 111
 - regular, 111
 - T-pyramid, 108
 - tree, 108
- Q**
- quadric surface model, 663
 - quadtrees, **109**, 225, 232, 335, 337, 354, 372, 731
 - qualitative vision, 588
 - quantization, 15
 - vector, 38
 - quench function, 704
- R**
- R-table, 216
 - RADAR, 628
 - radial distortion, 88
 - radiance, 4, 90, 92
 - spectral, 93
 - radiant flux, 89
 - radiometry, 89
 - random dot stereograms, 624
 - random forest, 442–448, 517–524, 841
 - random sample consensus, *see* RANSAC
 - range image, 628, 663
 - RANSAC, 477–481
 - receptive field, 143
 - reconstruction
 - morphological, 706, 709
 - projective, 605
 - rectification, 616, 617
 - rectified configuration, 615
 - rectilinear cameras, 615
 - reduce, 690
 - redundancy, 24, 79, 125
 - information, 722–727
 - reflectance, 13, 92
 - coefficient, 94
 - function, 94
 - map, 94
 - surface, 92
 - reflectance function, 631
 - reflection
 - body, 31
 - surface, 31
 - region, **17**, 19
 - concavity tree, 364
 - convex, 22
 - decomposition, 353, **370–371**
 - growing, 179
 - identification, 333–335
 - skeleton, 348, 353, 365, 367, 369–371
 - region adjacency graph, 106
 - region adjacency graph , 112
 - region map, 106
 - regional extreme, 705
 - registration, *see* image, registration
 - relation
 - neighborhood, 102
 - spatial, 102
 - relational structure, 389, 413, 419, 421, 465
 - relaxation, 190, **525–530**, 533, 542, 552, 626, 633, 796, 797, 799
 - discrete, 525, 527
 - probabilistic, 527, 529
 - remote sensing, 120, 186, 187, 505, 532, 724
 - representation, 585
 - 3D, 586
 - complete, 656
 - geometric, 101
 - iconic image, 101
 - intermediate, 13, 100
 - level of, 100
 - multi-view, 670
 - relational model, 101
 - segmented image, 101
 - skeleton, 662
 - unique, 656
 - reproducibility, 241
 - reproduction, 428, 429
 - resolution
 - radiometric, 14
 - spatial, 8, 14
 - spectral, 14
 - temporal, 8
 - time, 14
 - restoration, 162–165
 - deterministic, 162
 - geometric mean filtering, 165
 - inverse filtering, 163

- power spectrum equalization, 165
- stochastic, 162
- Wiener filtering, 164
- retina
 - cone, 32
 - rod, 32
- rigidity constraint, 646
- rim, 653
- rod, 32
- rotating mask, 129
- rotational movement, 646, 648, 794
- run length coding, **105**, 334, 372, 730
- S**
- sampling, 14–15, 62–65
 - importance, 838
 - interval, 62
 - point, 14
- saturation, 37
- scale, 143–147, 312, 330, 586, 747–749, 755, 764
- Scale Invariant Feature Transform, *see* SIFT
- scale-space, 143, 330, 345, 346, 369, 374, 624
- segmentation, 7, 178–232, 467
 - border detection, 197–204, 217
 - border tracing, 191–196, 208
 - extended, 193–195
 - inner, 191, 192
 - outer, 191–194
 - classification-based, 503–506
 - complete, 7, 178, 179, 218
 - dynamic programming, 206–210
 - edge thresholding, 188
 - edge-based, 179, **187–219**, 232
 - evaluation, 236–241
 - border positioning errors, 238–239
 - mutual overlap, 237–238
 - STAPLE, 241
 - supervised, 237–239
 - unsupervised, 240–241
 - fuzzy connectivity, 280–288
 - global, 179
 - Hough transform, 210–217
 - generalized, 215, 217
 - match-based, 235
 - mean shift, 255–262
 - morphological, 713
 - multi-thresholding, 182
 - partial, 7, 178, 187, 188, 218
 - region construction
 - from borders, 218
 - from partial borders, 219
 - region growing, 179, 220–232, 531, 534, 542
 - color image, 220
 - merging, 193, 221–224
 - over-growing, 232
 - semantic, 531–534
 - split-and-merge, 225
 - splitting, 222, 224, 225
 - under-growing, 232
 - region-based, 179, 195, 220–229
 - semantic, 531
 - region growing, 532–534
 - texture, 431
 - thresholding, **179–187**, 219, 223, 225
 - minimum error, 183
 - multi-spectral, 186
 - Otsu, 183, 184
 - p-tile, 181, 188
 - tree, 225, 227
 - watersheds, 229, 534
- self-occlusion, 621
- semantic net, 389, 465, 524
- semantics, 4, 386, 389, 527, 531, 535, 542
- sensor
 - CCD, 41
 - photo-emission, 40
 - photosensitive, 40
- separable
 - linearly, 391
- separable filter, 127
- set difference, 685
- shading, 631
- Shannon sampling theorem, 14, 63
- shape, 329–373, 655
 - class, 330, 373
 - description
 - area, 337, 342, **354–355**, 356–358, 371
 - bending energy, 339
 - border length, 337, 339

- chord distribution, 340
- compactness, 193, 353, **357**, 537
- contour-based, 330, 332, **335–353**
- convex hull, 360, 361, 364
- cross ratio, 350
- curvature, 332, **337**, 343–348, 370
- direction, 353, **357**
- eccentricity, 356
- elongatedness, 330, 353, **356**, 357, 369
- Euler's number, 355
- external, 330, 332
- Fourier descriptors, 339, **341–342**
- graph, 353, **365**
- height, 355
- internal, 330, 332
- invariants, 349, 350, 352
- moments, 342, 348, 357, **358–360**, 371
- moments, area-based, 360
- moments, contour-based, 360
- perimeter, 193, 222, 223, 332, **337**
- polygonal, **343–345**, 346, 371
- projection-invariant, 331, 349
- projections, 355
- rectangularity, 353, **357**
- region concavity tree, 364
- region-based, 330, 332, 348, **353–373**
- run length, 334, 372
- segment sequence, 339, **343**
- signature, 339
- sphericity, 357
- spline, **346–348**, 817
- statistical, 330, 353
- syntactic, 329, 330, 344, 371
- width, 355
- primitive, 370
- shape from
 - contour, 653
 - de-focus, 653
 - focus, 652
 - motion, 644–651
 - optical flow, 648
 - shading, 631–634, 652
 - stereo, 621–627
 - texture, 651–652
 - vergence, 653
 - X, 586, 644
- sharpening, 135
- shrink, 690
- SIFT, 474–477, 479
 - key location, 474
- sifting, 51
- signal, 11
- signal-to-noise ratio, 29
- silhouette, 654
- simple contiguous, 18
- simulated annealing, 430, 431
- Simultaneous Localization and Mapping, *see* SLAM
- singular point, 633
- singular value decomposition, 73, 148
- skeleton, 348, 353, 365, 367, 369–371, 662, 684, 699–702
 - by influence zones, 707
 - by maximal balls, 700
- skew, 122
- SKIZ, skeleton by influence zones, 707
- SLAM, 815
- slant, 651
- smart snake, 487
- smoothing, 125–127
 - averaging, 125
 - averaging according to inverse gradient, 129
 - averaging with limited data validity, 128
 - edge preserving, 125
 - Gaussian, 345
 - non-linear mean filter, 133
 - order statistics, 132
 - rank filtering, 132
 - rotating mask, 129
- snake, 218, **263–273**, 789
 - B-snakes, 268
 - finite difference, 268
 - finite element, 268
 - Fourier, 268
 - gradient vector flow, 268–272
 - growing, 266
 - GVF, 268–272

- Hermite polynomial, 268
 - united, 268
 - SNR, 29
 - spatial angle, 90
 - spatial element, 281
 - spectral
 - BRDF, 93
 - density, 81
 - irradiance, 93
 - radiance, 93
 - spectrophotometer, 31
 - spectrum, 81
 - frequency, 150, 151, 711
 - granulometric, 711
 - power, 31
 - spel, *see* spatial element
 - spline, 264
 - STAPLE, 241
 - state space search, 222
 - stereo
 - correspondence, 431, 645
 - photometric, 635
 - vision, 233, 609, 645
 - stereopsis, 609
 - STFT—short time FT, 58
 - stochastic process, 79–81
 - stationary, 81
 - uncorrelated, 80
 - structure from motion theorem, 646
 - structured light, 628
 - structuring element (morphological), 685
 - sufficient statistics, 562
 - super-quadrics, 660
 - supergrid, 193, 222, 223
 - superposition principle, 51
 - support vector, 400, 402
 - support vector machine, 392, 400–405, 762
 - kernel function, 403
 - margin, 400, 441
 - surface
 - coupled, 303
 - detection, 288
 - multiple, 308, 310, 311
 - optimal, 303, 317
 - single, 306, 308
 - free-form, 655
 - reflection, 31
 - SVD, *see* singular value decomposition
 - SVM, *see* support vector machine
 - sweep representations, 661
 - symbol
 - non-terminal, 415
 - terminal, 415
 - syntactic analysis, 415–420, 770–772
 - syntax, 4, 386
 - system approach, 143
 - system theory, 584
- T**
- telecentric lens, 85
 - template matching, 233–236, 474, 513
 - tensor
 - multiview, 605
 - terahertz, 1, 7
 - test set, 397
 - texel, 651, 652, 747
 - texton, 750, 755
 - texture, 178, 179, 181, 187, 221, 431, 503, 506, 509, 651, **747–776**, 830
 - Brodatz, 751, 760, 761, 767
 - coarse, 747, 749–752, 755, 769
 - description
 - auto-correlation, 750
 - autocorrelation, 769
 - autoregression model, 769
 - chain grammar, 770
 - co-occurrence, 752, 753
 - discrete transform, 751
 - edge frequency, 754
 - fractal, 762, 770
 - grammar, 770
 - graph grammar, 772, 773, 775
 - hybrid, 749, 769, 775–776
 - Laws' energy measures, 757
 - morphology, 768
 - multiscale, 764–768
 - optical transform, 751
 - peak and valley, 769
 - primitive grouping, 773
 - primitive length, 755
 - run length, 755
 - shape rules, 770
 - statistical, 749–769, 773, 775
 - syntactic, 749, 769–775

- texture properties, 754
- texture transform, 768
- wavelet frames, 764–768
- wavelets, 761, 764–768
- deterministic tourist walk, 769
- element, 747
- fine, 747, 749–752, 755, 769
- gradient, 651, 652
- hierarchical, 773
- LBP, 757–762, 776
 - uniform, 759
- local ternary patterns, 762
- measure
 - uniformity, 759
- primitive, 219, 651, **747**, 749, 769, 770, 772, 774, 775
- segmentation, 755, 768, 773–775
- strong, 749, 775
- structure, 747, 749
- synthesis, 770, 771
- template
 - microstructure, 759
- tone, 747, 749, 776
- weak, 749, 775
- theory, computational, 585
- thickening, 684, 701–702
 - sequential, 702
- thin lens, 83
- thinning, 353, 365–367, 369, 684, 701–702
 - sequential, 701
- threshold, 179
 - optimal, 183
 - Otsu, 183, 184
 - selection, 179, 181
 - optimal, 183
- thresholding, 118, 282, 554
 - adaptive, 179
 - optimal, 183
 - with hysteresis, 146
- TIFF, 734
- tilt, 651
- tolerance interval, 343
- top surface (morphological), 694
- top-down approach, 418, 419, 467, 468
- topology, 22
 - discrete, 19
- tracking, 814
- kernel, 820–826
- mean shift, 820–826
- Predator, 840, 842
- semi-supervised, 840–843
- TLD, 840–843
- training set, 393, **396–397**, 398, 400, 406, 410, 420, 421, 482, 503, 508
- transformation
 - affine, 122
 - anti-extensive, 691
 - bilinear, 121
 - binary, 727
 - brightness correction, 117
 - cosine, 727, 732
 - discrete wavelet, 69
 - distance, 19, 369
 - fast wavelet, 69
 - Fourier, 14, 19, 53–64, 77, 125, 135, 148–150, 152, 163, 164, 234, 330, 341, 342, 549, 727, 751
 - inverse, 59
 - Gabor, 761, 764
 - geodesic, 707
 - geometric, 120–124
 - brightness interpolation, 123
 - change of scale, 122
 - pixel co-ordinate, 121
 - rotation, 122
 - skewing, 122
 - gray-scale, 117–120
 - histogram equalization, 118–120
 - logarithmic, 120
 - pseudo-color, 120
 - thresholding, 118
 - Haar, 68
 - Hadamard, 79, 727, 732
 - Hadamard-Haar, 78
 - hat, 124, 141
 - homotopic, 699
 - Hotelling, 74
 - Hough, 78, **210–217**, 348, 652
 - increasing, 689, 691
 - inverse, 692
 - Karhunen-Loève, 74, 724
 - morphological, 686
 - Paley, 78
 - pixel brightness, 117–120

870 Index

- pixel co-ordinate, 121
- projective, 592
- Radon, 77–78, 211, 355
- recursive block coding, 727
- reversible component, 736
- rubber sheet, 22
- sine, 78
- Slant-Haar, 78
- symmetric axis, 369
- top hat, 698
- Walsh, 78, 727
- wavelet, 66, 764
- z, 764
- translation, morphological, 686
- translational movement, 646, 648, 801, 803
- transmission
 - progressive, 731, 732
 - smart, 732
- tree
 - decision, 443
 - interval, 345
 - pruning, 202, 418
- triangulation, 603
- trichromacy, 32
- trifocal tensor, 620
- U**
 - ultimate erosion, 705
 - umbra (morphological), 694
 - uncertainty principle, 57
 - unit ball, 700
 - unsharp masking, 135
 - unsupervised learning, 406
 - upper semi-continuity, 687
- V**
 - vanishing point, 609, 652, 675
 - vector quantization, 38
 - velocity
 - field, 789
 - computation, 808
 - smoothness constraint, 795, 797
 - vector, 788
 - vergence, 653
 - vertex, 656
 - view
 - topographic (morphology), 694
 - viewing space, 670
 - viewing sphere, 670
 - viewpoint, 13
 - vignetting, 91
 - vision
 - active, 648
 - stereo, 233, 609, 645
 - view-based, 670
 - VISIONS, 473, 474
 - visual
 - potential, 671
 - system human, 25, 586, 609, 646, 648, 732, 769
 - Viterbi algorithm, 206, 546, 548
 - volumetric model, 658
 - volumetric primitives, 587
 - voxel, **15**, 185, 658
- W**
 - watersheds, 229, 714
 - wavelet, 66, 67, 764–768
 - energy
 - signature, 766
 - frames, 764–768
 - discrete, 764
 - mother, 67
 - packet, 72
 - weighted graph, 106
 - white balancing, 40
 - white noise, 28
 - Wiener filtering, 164
- X**
 - XBox, 517
 - Kinect, 517
- Z**
 - Zadeh operators, 434
 - zero-crossing, 136, **139**, 145, 345, 586, 624, 755
 - Zuniga–Haralick operator, 157, 805

