Eye Gaze Correction with Stereovision for Video-Teleconferencing

Ruigang Yang*[1] and Zhengyou Zhang[2]

[1] Dept. of Computer Science, University of North Carolina at Chapel Hill, North Carolina, USA
ryang@cs.unc.edu,
[2] Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
zhang@microsoft.com

Abstract. The lack of eye contact in desktop video teleconferencing substantially reduces the effectiveness of video contents. While expensive and bulky hardware is available on the market to correct eye gaze, researchers have been trying to provide a practical software-based solution to bring video-teleconferencing one step closer to the mass market. This paper presents a novel approach that is based on stereo analysis combined with rich domain knowledge (a personalized face model). This marriage is mutually beneficial. The personalized face model greatly improved the accuracy and robustness of the stereo analysis by substantially reducing the search range; the stereo techniques, using both feature matching and template matching, allow us to extract 3D information of objects other than the face and to determine the head pose in a much more reliable way than if only one camera is used. Thus we enjoy the versatility of stereo techniques without suffering from their vulnerability. By emphasizing a 3D description of the scene on the face part, we synthesize virtual views that maintain eye contact using graphics hardware. Our current system is able to generate an eye-gaze corrected video stream at about 5 frames per second on a commodity PC.
Keywords: Stereoscopic vision, Eye-gaze correction, Structure from motion.

1 Introduction

Video-teleconferencing, a technology enabling communicating with people face-to-face over remote distances, does not seem to be as widespread as predicted. Among many problems faced in video-teleconferencing, such as cost, network bandwidth, and resolution, the lack of eye-contact seems to be the most difficult one to overcome[18]. The reason for this is that the camera and the display screen cannot be physically aligned in a typical desktop environment. It results in unnatural and even awkward interaction. Special hardware using half-silver mirrors has been used to address this problem. However this arrangement is bulky and the cost is substantially high. What's more, as a piece of dedicated equipment, it does not fit well to our familiar computing environment, thus its usability is greatly reduced. We aim to address the eye-contact problem by synthesizing videos as if they were taken from a camera behind the display screen, thus to establish natural eye-contact between video-teleconferencing participants without using any kind

* This work was mainly conducted while the first author was at Microsoft Research as a summer intern.

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2351, pp. 479–494, 2002.
© Springer-Verlag Berlin Heidelberg 2002

Microsoft Exhibit 1021 - Page 1
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

of special hardware. The approach we take involves three steps: pose tracking, view matching, and view synthesis. We use a pair of calibrated stereo cameras and a personalized face model to track the head pose in 3D. The use of strong domain knowledge (a personalized face model) and a stereo camera pair greatly increase the robustness and accuracy of the 3D head pose tracking. The stereo camera pair also allows us to match parts not modelled in the face model, such as the hands and shoulders, thus providing wider coverage of the subject. Finally, the results from the head tracking and stereo matching are combined to generate a virtual view. Unlike other methods that only "cut and paste" the face part of the image, our method generates natural looking and seamless images, as shown in figure 1.

Fig. 1. Eye-gaze Correction: The two images shown on the left are taken from a pair of stereo cameras mounted on the top and bottom sides of a monitor; the image shown above is a synthesized virtual view that preserves eye-contact.

There have been attempts to address the eye-contact problem using a single camera with a face model [12],[10], or using dense stereo matching techniques[19], [15]. We will discuss these approaches in more details in Section 2. With a single camera, we found it difficult to maintain both the real-time requirement and the level of accuracy we want with head tracking. Existing model-based monocular head tracking methods [11], [4], [2], [7], [1] either use a simplistic model so they could operate in real time but produce less accurate results, or use some sophisticated models and processing to yield highly accurate results but take at least several seconds to compute. A single-camera configuration also has difficulties to deal with occlusions. Considering these problems with a monocular system, we decided to adopt a stereo configuration. The important epipolar constraint in stereo allows us to reject most outliers without using expensive robust estimation techniques, thus keeping our tracking algorithm both robust and simple enough to operate in real-time. Furthermore, two cameras usually provide more coverage of the scene.

One might raise the question that why we do not use a dense stereo matching algorithm. We argue that, first, doing a dense stereo matching on a commodity PC in real time is difficult, even with today's latest hardware. Secondly and most importantly, a dense stereo matching is unlikely to generate satisfactory results due to the limitation on camera placement. Aiming at desktop video conferencing applications, we could only put the cameras around the frame of a display monitor. If we put the cameras on the opposite edges of the display, given the normal viewing distance, we have to converge the cameras towards the person sitting in front of the desktop, and such a stereo system will have a long baseline. That makes stereo matching very difficult; even if we were able to get a perfect matching, there would still be a significant portion of the subject which is occluded in one view or the other. Alternatively, if we put the cameras close to

Microsoft Exhibit 1021 - Page 2
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

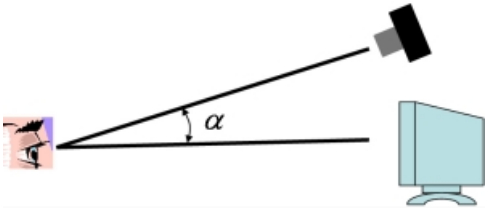


Fig. 2. Camera-Screen displacement causes the lose of eye-contact.

each other on the same edge of the monitor frame, the occlusion problem is less severe, but generalization to new distant views is poor because a significant portion of the face is not observed. After considering various aspects, we have decided to put one camera on the upper edge of the display and the other on the lower edge, and follow a model-based stereo approach to eye-gaze correction.

2 Related Works

In a typical desktop video-teleconferencing setup, the camera and the display screen cannot be physically aligned, as depicted in figure 2. A participant looks at the image on the monitor but not directly into the camera, thus she does not appear to make eye contact with the remote party. Research [23] has shown that if the divergence angle (α) between the camera and the display is greater than five degrees, the loss of eye-contact is noticeable. If we mount a small camera on the side of a 21-inch monitor, and the normal viewing position is at 20 inches away from the screen, the divergence angle will be 17 degrees, well above the threshold at which the eye-contact can be maintained. Under such a setup, the video loses much of its communication value and becomes un-effective compared to telephone.

Several systems have been proposed to reduce or eliminate the angular deviation using special hardware. They make use of half-silvered mirrors or transparent screens with projectors to allow the camera to be placed on the optical path of the display. A brief review of these hardware-based techniques has been given in [14]. The expensive cost and the bulky setup prevent them to be used in a ubiquitous way.

On the other track, researchers have attempted to create eye-contact using computer vision and computer graphics algorithms. Ott et al. [19] proposed to create a virtual center view given two cameras mounted on either side of the display screen. Stereoscopic analysis of the two camera views provides a depth map of the scene. Thus it is possible to “rotate" one of the views to obtain a center virtual view that preserves eye contact. Similarly, Liu et al. [15] used a trinocular stereo setup to establish eye contact. In both cases, they perform dense stereo matching without taking into account the domain knowledge. While they are generic enough to handle a variety of objects besides faces, they are likely to suffer from the vulnerability of brute-force stereo matching. Furthermore, as discussed in the previous section, we suspect that direct dense stereo matching is unlikely to generate satisfactory results due to the constraint of camera placement imposed by the size of the display monitor – a problem that may be less severe back in the early 90’s, when the above two algorithms were proposed .

Cham and Jones at Compaq Cambridge Research Laboratory [6] approached this problem from a machine learning standpoint. They first register a 2D face model to

Microsoft Exhibit 1021 - Page 3
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

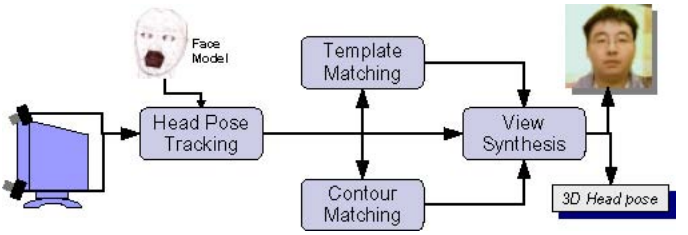


Fig. 3. The components of our eye-gaze correction system

the input image taken from a single camera, then morph the face model to the desired image. The key is to learn a function that mapps the registered face model parameters to the desired morphed model parameters [12]. They achieve this by non-linear regression from sufficient instances of registered-morphed parameter pairs which are obtained from training data. As far as we know, their research is still in a very early stage, so it is not clear if this approach is capable of handling dramatic facial expression changes. Furthermore, they only deal with the face part of the image – the morphed face image is superimposed on the original image frame, which sometime leads to errors near the silhouettes due to visibility changes.

The GazeMaster project at Microsoft Research [10] uses a single camera to track the head orientation and eye positions. Their view synthesis is quite unique in that they first replace the human eyes in a video frame with synthetic eyes gazing in the desired direction, then texture-map the eye-gaze corrected video frame to a generic rigid face model rotated to the desired orientation. The synthesized photos they published look more like avatars, probably due to the underlying generic face model. Another drawback is that, as noted in their report, using synthetic eyes sometime inadvertently changes the facial expression as well.

From a much higher level, this GazeMaster work is similar to our proposed approach, in the sense that they both use strong domain knowledge (a 3D face model) to facilitate the tracking and view synthesis. However, our underlying algorithms, from tracking to view synthesis, are very different from theirs. We incorporate a stereo camera pair, which provides the important epipolar constraint that we use throughout the entire process. Furthermore, the configuration of our stereo camera provides much wider coverage of the face, allowing us to generate new distant views without having to worry about occlusions.

3 System Overview

Figure 3 illustrates the block diagram of our eye-gaze correction system. We use two digital video cameras mounted vertically, one on the top and the other on the bottom of the display screen. They are connected to a PC through 1394 links. The cameras are calibrated using the method in [24]. We choose the vertical setup because it provides wider coverage of the subject and higher disambiguation power in feature matching. Matching ambiguity usually involves symmetric facial features such as eyes and lip contours aligned horizontally. The user's personalized face model is acquired using a rapid face modelling tool [16]. Both the calibration and model acquisition require little human interaction, and a novice user can complete these tasks within 15 minutes.

Microsoft Exhibit 1021 - Page 4
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

With these prior knowledge, we are able to correct the eye-gaze using the algorithm outlined as follows:

```
1. Background model acquisition
2. Face tracker initialization
3. For each image pair, perform
    - Background subtraction
    - Temporal feature tracking in both images
    - Updating head pose
    - Correlation-based stereo feature matching
    - Stereo silhouette matching
    - Hardware-assisted view synthesis
```

Currently, the only manual part of the system is the face tracker initialization which requires the user to interactively select a few markers. We are currently working on automatic initialization.

The tracking subsystem includes a feedback loop that supplies fresh salient features at each frame to make the tracking more stable under adversary conditions, such as partial occlusions and facial expression changes. Furthermore, an automatic tracking recovery mechanism is also implemented to make the whole system even more robust over extended period of time. Based on the tracking information, we are already able to manipulate the head pose by projecting the live images onto the face model. However, we also want to capture the subtleties of facial expressions and other foreground objects, such as hands and shoulders. So we further conduct correlation-based feature matching and silhouette matching between the stereo images. All the matching information, together with the tracked features, is used to synthesize a *seamless* virtual image that looks as if it were taken from a camera behind the display screen. We have implemented the entire system under the MS Windows environment. Without any effort spending on optimizing the code, our current implementation runs about 4-5 frames per second on a single CPU 1 GHz PC.

4 Stereo 3D Head Pose Tracking

Our stereo head pose tracking problem can be stated as follows:

Given (i) a set of triplets $S = \{(\mathbf{p}, \mathbf{q}, \mathbf{m})\}$ at time t, where $\mathbf{p}$ and $\mathbf{q}$ are respectively points in the upper (first) and the lower (second) camera images, and $\mathbf{m}$ is their corresponding point in the face model, and (ii) a pair of images from the stereo cameras at time $t+1$,

determine (i) $S' = \{(\mathbf{p}', \mathbf{q}', \mathbf{m})\}$ at time $t+1$ where $\mathbf{p}'$ and $\mathbf{q}'$ are the new positions of $\mathbf{p}$ and $\mathbf{q}$, and (ii) compute the head pose, so that the projections of $\mathbf{m}$ in time $t+1$ are $\mathbf{p}'$ and $\mathbf{q}'$ in the stereo image pair, respectively.

We use the KLT tracker to track feature points $\mathbf{p}, \mathbf{q}$ from time t to $t+1$ [22]. Note that there is one independent feature tracker for each camera, thus we need apply the epipolar constraint to remove any stray point. The epipolar constraint states that if a point $\mathbf{p} = [u, v, 1]^T$ (expressed in homogeneous coordinates) in the first image and a

Microsoft Exhibit 1021 - Page 5
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

point $\mathbf{q} = [u', v', 1]^T$ in the second image correspond to the same 3D point $\mathbf{m}$ in the physical world, they must satisfy the following equation:

$$\mathbf{q}^T \mathbf{F} \mathbf{p} = 0 \tag{1}$$

where $\mathbf{F}$ is the fundamental matrix that encodes the epipolar geometry between the two images [9]. In fact, $\mathbf{Fp}$ defines the epipolar line in the second image, thus Equation (1) means that the point $\mathbf{q}$ must pass through the epipolar line $\mathbf{Fp}$ and vice versa.

In practice, due to camera noise and inaccuracy in camera calibration and feature localization, we define a band of uncertainty along the epipolar line. For every triplet $(\mathbf{p}', \mathbf{q}', \mathbf{m})$, if the distance from $\mathbf{q}'$ to the $\mathbf{p}'$s epipolar line is greater than a certain threshold, this triplet will be discarded. We use a distance threshold of three pixels in our experiment.

After we have removed all the stray points that violate the epipolar constraint, we update the head pose, represented by a 3×3 rotational matrix $\mathbf{R}$ and a 3D translation vector $\mathbf{t}$, so that the sum of re-projection errors of $\mathbf{m}$ to $\mathbf{p}'$ and $\mathbf{q}'$ is minimized. The re-projection error e is defined as

$$e = \sum_i \| \mathbf{p}'_i - \phi(\mathbf{A}^0(\mathbf{R}\mathbf{m}_i + \mathbf{t}) \|^2 + \| \mathbf{q}'_i - \phi(\mathbf{A}^1[\mathbf{R}^{10}(\mathbf{R}\mathbf{m}_i + \mathbf{t}) + \mathbf{t}^{10}]) \|^2 \tag{2}$$

where $\phi(\cdot)$ represents the standard pinhole projection, $\mathbf{A}^0$ and $\mathbf{A}^1$ are the cameras' intrinsic parameters, and $(\mathbf{R}^{10}, \mathbf{t}^{10})$ is the transformation from the second camera's coordinate system to the first camera's. Solving $(\mathbf{R}, \mathbf{t})$ by minimizing (2) is a nonlinear optimization problem. We can use the head pose from time t as the initial guess and conduct the minimization by means of, for example, the Levenberg-Marquardt algorithm.

After the head pose is determined, we replenish the matched set S' by adding more good feature points selected using the criteria in [22]. A good feature point is a point with salient textures in its neighborhood. We must be careful not to add feature points in the non-rigid parts of the face, such as the mouth region. To do so, we define a bounding box around the tip of the nose that covers the forehead, eyes, and nose region. Any good feature points outside this bounding box will not be added to the matched set. However, they will be used in the next stereo matching stage, which we will discuss in Section 5.2.

The replenish scheme greatly improves the robustness of the tracking algorithm. Our experiments have shown that our tracking can keep tracking under large head rotations and dramatic facial expression changes.

4.1 Tracker Initialization and Auto-Recovery

The tracker needs to know the head pose at time 0 to start tracking. We let the user interactively select seven landmark points in each image, from which the initial head pose can be determined. The initial selection is also used for tracking recovery when the tracker loses tracking. This may happen when the user moves out of the camera's field of view or rotates her head away from the cameras. Fortunately, for our video-conferencing application, we could just send one of the video streams un-modified for these cases. When she turns back to look at the screen, we do need to continue tracking with no human intervention, which requires automatic recovery of the head pose. During the tracking recovery process, the initial set of landmark points is used as templates to find the best match in the current image. When a match with a high confidence value is found, the

Microsoft Exhibit 1021 - Page 6
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

tracker continues the normal tracking. Our recovery scheme is effective because unlike most other tracking applications, we only need to track the user when she is looking at the display window. This is exactly the scenario when the initial landmark templates are recorded.

Furthermore, we also activate the auto-recovery process whenever the current head pose is close to the initial head pose. This prevents the tracker from drifting. A more elaborated scheme that uses multiple templates from different head poses can further improve the effectiveness of automatic tracker reset. This can be further extended to a non-parametric description of head poses that can self-calibrate over time.

5 Stereo View Matching

The result from the 3D head pose tracking gives a set of good matches within the *rigid* part of the face between the stereo pair. To generate convincing and photo-realistic virtual views, we need to find more matching points over the entire foreground images, especially along the contour and the non-rigid parts of the face. We incorporate both feature matching and template matching to find as many matches as possible. During this matching process, we use the reliable information obtained from tracking to constrain the search range. In areas where such information is not available, such as hands and shoulders, we relax the search threshold, then apply the disparity gradient limit to remove false matches. To facilitate the matching (and later view synthesis in Section 6), we rectify the images using the technique described in [17], so that the epipolar lines are horizontal.

5.1 Disparity and Disparity Gradient Limit

Before we present the details of our matching algorithm, it is helpful to define disparity, disparity gradient, and the important principle of disparity gradient limit, which will be exploited through out the matching process.

Disparity is well defined for parallel cameras (i.e., the two image planes are the same) [9], and this is the case because we perform stereo rectification such that the horizontal axes are aligned in both images. Given a pixel (u, v) in the first image and its corresponding pixel (u', v') in the second image, disparity is defined as $d = u' - u$ ($v = v'$ as images have been rectified). Disparity is inversely proportional to the distance of the 3D point to the cameras. A disparity of 0 implies that the 3D point is at infinity.

Consider now two 3D points whose projections are $\mathbf{m}_1 = [u_1, v_1]^T$ and $\mathbf{m}_2 = [u_2, v_2]^T$ in the first image, and $\mathbf{m}'_1 = [u'_1, v'_1]^T$ and $\mathbf{m}'_2 = [u'_2, v'_2]^T$ in the second image. Their disparity gradient is defined to be the ratio of their difference in disparity to their distance in the cyclopean image, i.e.,

$$DG = \left| \frac{d_2 - d_1}{u_2 - u_1 + (d_2 - d_1)/2} \right| \tag{3}$$

Experiments in psychophysics have provided evidence that human perception imposes the constraint that the disparity gradient DG is upper-bounded by a limit K. The limit $K = 1$ was reported in [5]. The theoretical limit for opaque surfaces is 2 to ensure that the surfaces are visible to both eyes [20]. Also reported in [20], less than 10% of world surfaces viewed at more than 26cm with 6.5cm of eye separation will present with disparity gradient larger than 0.5. This justifies use of a disparity gradient limit well below the theoretical value (of 2) without imposing strong restrictions on the world

Microsoft Exhibit 1021 - Page 7
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

surfaces that can be fused by the stereo algorithm. In our experiment, we use a disparity gradient limit of 0.8 ($K = 0.8$).

5.2 Feature Matching Using Correlation

For unmatched good features in the first (upper) image, we try to find corresponding points, if any, in the second (lower) image by template matching. We use normalized correlation over a 9×9 window to compute the matching score. The disparity search range is confined by existing matched points from tracking, when available.

Combined with matched points from tracking, we build a sparse disparity map for the first image and use the following procedure to identify potential outliers (false matches) that do not satisfy the disparity gradient limit principle. For a matched pixel $\mathbf{m}$ and a neighboring matched pixel $\mathbf{n}$, we compute their disparity gradient between them using (3). If $DG \leq K$, we register a vote of good match for $\mathbf{m}$; otherwise, we register a vote of bad match for $\mathbf{m}$. After we have counted for every matched pixel in the neighborhood of $\mathbf{m}$, we tally the votes. If the "good" votes are less than the "bad" votes, $\mathbf{m}$ will be removed from the disparity map. This is conducted for every matched pixel in the disparity map; the result is a disparity map that conforms to the principle of disparity gradient limit.

5.3 Contour Matching

Template-matching assumes that corresponding images patches present some similarity. This assumption may be wrong at occluding boundaries, or object contours. Yet object contours are very important cues for view synthesis. The lack of matching information along object contours will result in excessive smearing or blurring in the synthesized views. So it is necessary to include a module that extracts and matches the contours across views in our system.

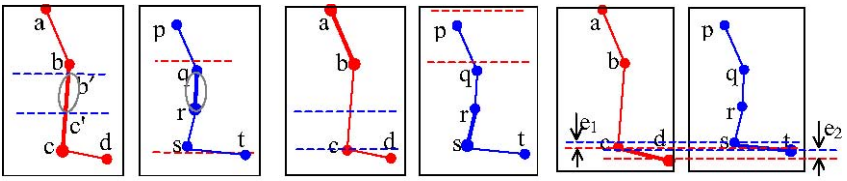
The contour of the foreground object can be extracted after background subtraction. It is approximated by polygonal lines using the Douglas-Poker algorithm[8]. The control points on the contour are further refined to sub-pixel accuracy using the "snake" technique[13]. Once we have two polygonal contours, denoted by $P = \{\mathbf{v}_i | i = 1..n\}$ in the first image and $P' = \{\mathbf{v}'_i | i = 1..m\}$ in the second image, we use the dynamic programming technique (DP) to find the global optimal match across them.

Since it is straightforward to formulate contour matching as a dynamic programming problem with states, stage, and decisions, we will only discuss in detail the design of the cost functions (The reader is referred to Bellman's book about dynamic programming techniques [3]). There are two cost functions, the *matching* cost and the *transition* cost. The matching cost function $C(i, j)$ measures the "goodness" of matching between segment $V_i = \overline{\mathbf{v}_i \mathbf{v}_{i+1}}$ in P and segment $V'_j = \overline{\mathbf{v}'_i \mathbf{v}'_{i+1}}$ in P'. The lower the cost, the better the matching. The transition cost function $W(i, j | i_0, j_0)$ measures the smoothness from segment V_{i_0} to segment V_i, assuming that (V_i, V'_j) and (V_{i_0}, V'_{j_0}) are matched pairs of segments. Usually, V_i and V_{i_0} are continuous segments, i.e., $\|i_0 - i\| \leq 1$. It penalize for matches that are out of order. The scoring scheme of DP, formulated as a forward recursion function, is then given by

$$\begin{aligned} M(i,j) = \min(\ & M(i-1,j-1) + C(i,j) + W(i,j|i-1,j-1), \\ & M(i,j-1) + C(i,j) + W(i,j|i,j-1), \\ & M(i-1,j) + C(i,j) + W(i,j|i-1,j)\ )\ . \end{aligned}$$

Microsoft Exhibit 1021 - Page 8
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

The Matching Cost. It takes into account the epipolar constraint, the orientation difference, and the disparity gradient limit.



(a) Two segments overlap (b) Two segments do not overlap (c) Segments almost parallel to the epipolar lines

Fig. 4. Applying the epipolar constraint to contour matching.

- **The epipolar constraint**: We distinguish three configurations, as shown in Figure 4 where the red line is the contour in the first (upper) image; the blue line is the contour in the second (lower) image. The dotted lines are the corresponding epipolar lines. In Figure 4(a), segment $\overline{bc}$ and segment $\overline{qr}$ are being matched, and $C_e = 0$. The epipolar constraint limits $\overline{qr}$ corresponding to segment $\overline{b'c'}$, instead of $\overline{bc}$. In Figure 4(b), the epipolar constraint tells that segment $\overline{ab}$ cannot match segment $\overline{rs}$ because there is no overlap. In that case, a sufficiently large cost ($T_{highcost}$) is assigned to this match. When the orientation of at least one line segment is very close to that of epipolar lines, intersection of the epipolar line with the line segment cannot be computed reliably. In that case, the cost is the average inter-epipolar distance ($d_e = (e_1 + e_2)/2$), as illustrated in the figure. In summary, the epipolar constraint cost for a pair of segment (V_i, V'_j) is

$$C_e = \begin{cases} d_e & \text{if } V_i \text{ or } V'_j \text{ is close to horizontal lines;} \\ 0 & \text{if } V_i \text{ or } V'_j \text{ overlaps;} \\ T_{highcost} & \text{otherwise.} \end{cases} \tag{4}$$

- **The orientation difference**: It is defined as a power function of the orientation difference between the proposed matching segments (V_i, V'_j). Let a_i and a_j be orientation of V_i and V'_j, the orientation difference is

$$C_a = \left(\frac{|a_i - a_j|}{T_a}\right)^n \tag{5}$$

  where T_a is the angular difference threshold, and n is the power factor. We use $T_a = 30°$ and $n = 2$.
- **The disparity gradient limit**: It is similar to that used in template matching. However, we do not want to consider feature points in matching contour segments because the contour is on the occluding boundary, where the disparity gradient with respect to the matched feature points is very likely to exceed the limit. On the other hand, it is reasonable to assume that the disparity gradient limit will be upheld *between* the two endpoints of the segment. We adopt the disparity prediction model in [25]. That is, given a pair of matched points $(\mathbf{m}_i, \mathbf{m}'_i)$, the disparity of a point $\mathbf{m}$ is modeled as

$$d = d_i + D_i n_i \tag{6}$$

Microsoft Exhibit 1021 - Page 9
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

where $d_i = \|\mathbf{m}'_i - \mathbf{m}_i\|$, $D_i = \|\mathbf{m} - \mathbf{m}_i\|$, and $n_i \sim N(0, \sigma^2 I)$ with $\sigma = K/(2-K)$. A pair of matched segments contains two pairs of matched endpoints $(\mathbf{m}_s, \mathbf{m}'_s)$ and $(\mathbf{m}_e, \mathbf{m}'_e)$. We use $(\mathbf{m}_s, \mathbf{m}'_s)$ to predict the disparity of $\mathbf{m}_e$, and compute the variance of the "real" disparity from the predicted one. Similarly we also compute the variance of the predicted disparity of $\mathbf{m}_s$ using $(\mathbf{m}_e, \mathbf{m}'_e)$. As suggested in [25], the predicted variance should be less restrictive when the point being considered is away from the matched point, which leads to the following formulae:

$$\sigma_i = [\sigma_{max} - \sigma_{min}](1 - \exp(-D_i^2/\tau^2)) + \sigma_{min} \tag{7}$$

where the range $[\sigma_{min}, \sigma_{max}]$ and τ are parameters. We use $\sigma_{min} = 0.5, \sigma_{max} = 1.0,$ and $\tau = 30$.

Now we can finally write out the disparity gradient cost. Let $d_s = \|\mathbf{m}'_s - \mathbf{m}_s\|$, $d_e = \|\mathbf{m}'_e - \mathbf{m}_e\|$, $D_1 = \|\mathbf{m}_e - \mathbf{m}_s\|$, $D_2 = \|\mathbf{m}'_e - \mathbf{m}'_s\|$, and $\Delta d = d_e - d_s$; σ_e and σ_s are computed by plugging in D_1 and D_2 into (7); the disparity gradient cost is given by

$$C_d = \Delta d^2/\sigma_e^2 + \Delta d^2/\sigma_s^2 \ . \tag{8}$$

Combining all the above three terms, we have the final matching cost as:

$$C = \max\left(T_{highcost}, C_e + w_a C_a + w_d C_d\right) \tag{9}$$

where w_a and w_d are weighting constants. The match cost is caped by $T_{highcost}$. This is necessary to prevent any corrupted segment in the contour from contaminating the entire matching. In our implementation, $T_{highcost} = 20, w_a = 1.0,$ and $w_d = 1.0$.

The Transition Cost. In contour matching, when two segments are continuous in one image, we would prefer that their matched segments in the other image are continuous too. This is not always possible due to changes in visibility: some part of the contour can only be seen in one image. The transition cost (W) is designed to favor smooth matching from one segment to the next, while taking into account discontinuities due to occlusions. The principle we use is again the gradient disparity limit. For two consecutive segments V_i and V_{i+1} in P, the transition cost function is the same as the one used in matching cost – equation (8), except that the two pairs of matched points involved are now the endpoint of V_i and the starting point of V_{i+1} and their corresponding points in P'.

6 View Synthesis

From the previous tracking and matching stages, we have obtained a set of point matches and line matches that can be used to synthesize new views. Note that these matches contain not only the modeled face part, but also other foreground part such as hands and shoulders. This is yet another advantage of our stereovision-based approach. We could obtain a more complete description of the scene geometry beyond the limit of the face model. Treating these matches as a whole, our view synthesis methods can create seamless virtual imagery. We implemented and tested two methods for view synthesis. One is based on view morphing [21] and the other uses hardware-assisted multi-texture blending. The view morphing technique allows to synthesize virtual views along the path connecting the optical centers of the two cameras. A view morphing factor c_m controls the exact view position. It is usually between 0 and 1, whereas a value of 0 corresponds exactly to the first camera view, and a value of 1 corresponds exactly to the second

Microsoft Exhibit 1021 - Page 10
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

camera view. Any value in between represents a virtual viewpoint somewhere along the path from the first camera to the second.

In our hardware-assisted rendering method, we first create a 2D triangular mesh using Delaunay triangulation in the first camera's image space. We then offset each vertex's coordinate by its disparity modulated by the view morphing factor c_m, $[u'_i, v'_i] = [u_i + c_m d_i, v_i]$. The offset mesh is fed to the hardware render with two sets of texture coordinates, one for each camera image. Note that all the images and the mesh are in the rectified coordinate space. We need to set the viewing matrix to the inverse of the rectification matrix to "un-rectify" the resulting image to its normal view position. This is equivalent to the post-warp in view morphing. Thus the hardware can generate the final synthesized view in a single pass. We also use a more elaborate blending scheme, thanks to the powerful graphics hardware. The weight W_i for the vertex V_i is based on the product of the total area of adjacent triangles and the view-morphing factor, as

$$W_i = \frac{\sum S_i^1 * (1 - c_m)}{\sum S_i^1 * (1 - c_m) + \sum S_i^2 * c_m}; \tag{10}$$

where S_i^1 are the areas of the triangles of which V_i is a vertex, and S_i^2 are the areas of the corresponding triangles in the other image. By changing the view morphing factor c_m, we can use the graphics hardware to synthesize correct views with desired eye gaze in real-time. Because the view synthesis process is conducted by hardware, we can spare the CPU for more challenging tracking and matching tasks.

Comparing these two methods, the hardware-assisted method, aside from its blazing speed, generates crisper results if there is no false match in the mesh. On the other hand, the original view morphing method is less susceptible to bad matches, because it essentially uses every matched point or line segment to compute the final coloring of a single pixel, while in the hardware-based method only the three closest neighbors are used.

Regarding the background, it is very difficult to obtain a reliable set of matches since the baseline between the two views is very large, as can be observed in the two original images shown in Fig. 1. In this work, we do not attempt to model the background at all, but we offer two solutions. The first is to treat the background as unstructured, and add image boundary as matches. The result will be ideal if the background has a uniform color; otherwise, it will be fuzzy as shown in the synthesized view shown in Fig. 1. The second solution is to replace the background by anything appropriate. In that case, view synthesis is only performed for the foreground objects. In our implementation, we overlay the synthesized foreground objects on the image from the first camera. The results shown in the following section were produced in this way.

7 Experiment Results

We have implemented our proposed approach using C++ and tested with several sets of real data. Very promising results have been obtained. We will first present a set of sample images to further illustrate our algorithm, then we will show some more results from different test users. For each user, we built a personalized face model using a face modelling tool[16]. This process, which takes only a few minutes and requires no additional hardware, only needs to be done once per user. All the parameters in our algorithm are set to be the same for all the tests.

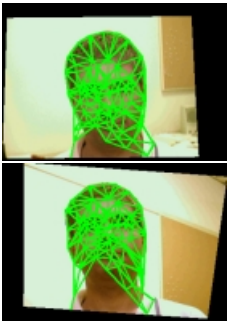
Microsoft Exhibit 1021 - Page 11
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

(a) The input image pair

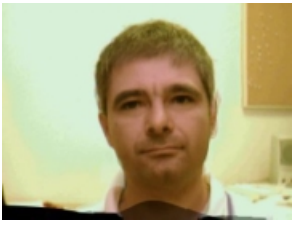
(b) Tracked feature points with epipolar line superimposed

(c) Extracted foreground contours

(d) Rectified images for stereo matching



(e) Delaunay triangulation over matched points



(f) Final synthesized view (uncropped)

Fig. 5. Intermediate results of our eye-gaze correction algorithm

Figure 5 shows the intermediate results at various stages of our algorithm. It starts with a pair of stereo images in Fig. 5(a); Fig. 5(b) shows the matched feature points, the epipolar lines of feature points in the first image are drawn in the second image. Fig. 5(c) shows the extracted foreground contours: the red one (typically a few pixels far away from the "true" contour) is the initial contour after background subtraction while the blue one is the refined contour using the “snake" technique. In Fig. 5(d) we show the rectified images for template matching. All the matched points form a mesh using Delaunay triangulation, as shown in Fig. 5(e). The last image (Fig. 5(f)) shows the synthesized virtual view. We can observe that the person appears to look down and up in the two original image but look forward in this synthesized view.

During our experiments, we captured all test sequences with resolution 320x240 at 30 frames per second. Our current implementation can only run at 4 to 5 frames per second. The results shown here are computed with our system in a “step-through" mode.

Microsoft Exhibit 1021 - Page 12
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

Except the manual initialization performed once at the beginning of the test sequences, the results are computed automatically without any human interaction.

Fig. 6. Sample results from the first test sequence. The top and bottom rows show the images from the top and bottom cameras. The middle row displays the synthesized images from a virtual camera located in the middle of the real cameras. The frame numbers from left to right are 108, 144, 167 and 191.

The first sequence (Figure 6) shows the viability of our system. Note the large disparity changes between the upper and lower camera images, making direct template-based stereo matching very difficult. However, our model-based system is able to accurately track and synthesize photo-realistic images under the difficult configuration, even with partial occlusions or oblique viewing angles.

Our second sequence is even more challenging, containing not only large head motions, but also dramatic facial expression changes and even hand waving. Results from this sequence, shown in Figure 7, demonstrated that our system is both effective and robust under these difficult conditions. Non-rigid facial deformations, as well as the subject's torso and hands, are not in the face model, yet we are still able to generate seamless and convincing views, thanks to our view matching algorithm that includes a multitude of stereo matching primitives (features, templates, and curves). Templates matching finds matching points, as many as possible, in regions where the face model does not cover, while contour matching preserves the important visual cue of silhouettes.

8 Conclusions

In this paper, we have presented a software scheme for maintaining eye contact during video-teleconferencing. We use model-based stereo tracking and stereo analysis to compute a partial 3D description of the scene. Virtual views that preserve eye contact are then synthesized using graphics hardware. In our system, modeled-based head tracking and

Microsoft Exhibit 1021 - Page 13
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

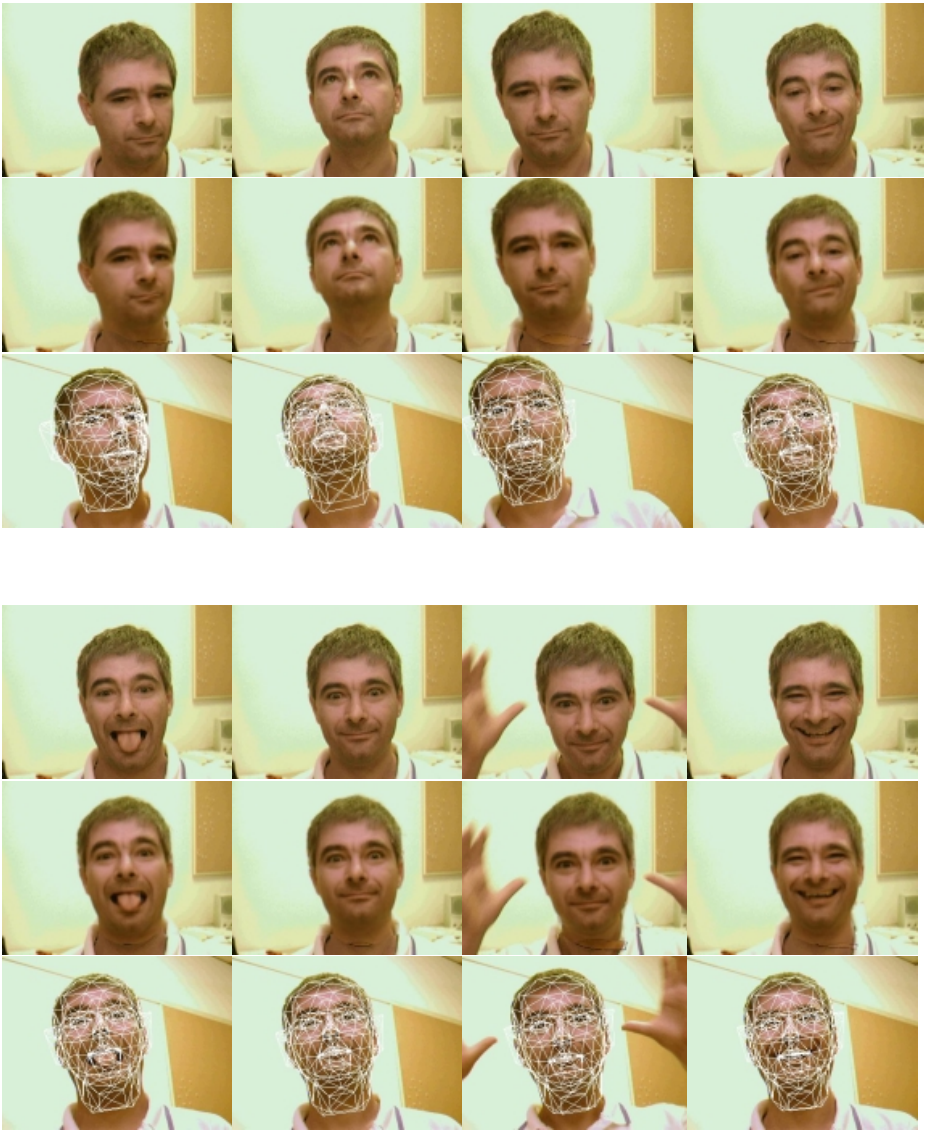


Fig. 7. Sample results from the second test sequence. The upper and lower rows are the original stereo images, while the middle rows are the synthesized ones. The triangular face model is overlayed on the bottom images. From left to right and from top to bottom, the frame numbers are 159, 200, 400, 577, 617, 720, 743, and 830.

Microsoft Exhibit 1021 - Page 14
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

stereo analysis work hand in hand to provide a new level of accuracy, robustness, and versatility that neither of them alone could provide. Experimental results have demonstrated the viability and effectiveness of our proposed approach.

While we believe that our proposed eye-gaze correction scheme represents a large step towards a viable video-teleconferencing system for the mass market, there are still plenty of rooms for improvements, especially in the stereo view matching stage. We have used several matching techniques and prior domain knowledge to find good matches as many as possible, but we have not exhausted all the possibilities. We believe that the silhouettes in the virtual view could be more clear and consistent across frames if we incorporate temporal information for contour matching. Furthermore, there are still salient curve features, such as hairlines and necklines, that sometimes go unmatched. They are very difficult to match using a correlation-based scheme because of highlights and visibility changes. We are investigating a more advanced curve matching technique.

References

1. A. Azarbayejani, B. Horowitz, and A. Pentland. Recursive Estimation of Structure and Motion Using the Relative Orientation Constraint. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 70–75, 1993.
2. S. Basu, I. Essa, and A. Pentland. Motion Regularization for Model-based Head Tracking. In *Proceedings of International Conference on Pattern Recognition*, pages 611–616, Vienna, Austria, 1996.
3. R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.
4. M. J. Black and Y. Yacoob. Tracking and Recognizing Rigid and Non-Rigid Facial Motions Using Local Parametric Model of Image Motion. In *Proceedings of International Conference on Computer Vision*, pages 374–381, Cambridge, MA, 1995.
5. P. Burt and B. Julesz. A gradient limit for binocular fusion. *Science*, 208:615–617, 1980.
6. T.J. Cham and M. Jones. Gaze Correction for Video Conferencing. Compaq Cambridge Research Laboratory, http://www.crl.research.digital.com/vision/interfaces/corga.
7. D. DeCarlo and D. Metaxas. Optical Flow Constraints on Deformable Models with Applications to Face Tracking. *International Journal of Computer Vision*, 38(2):99–127, July 2001.
8. D.H. Douglas and T.K. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature. *Canadian Cartographer*, 10(2):112–122, 1973.
9. O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.
10. J. Gemmell, C.L. Zitnick, T. Kang, K. Toyama, and S. Seitz. Gaze-awareness for Videoconferencing: A Software Approach. *IEEE Multimedia*, 7(4):26–35, October 2000.
11. T. Horprasert. Computing 3-D Head Orientation from a Monocular Image. In *International Conference of Automatic Face and Gesture Recognition*, pages 242–247, 1996.
12. Michael Jones. Multidimensional Morphable Models: A Framework for Representing and Matching Object Classes. *International Journal of Computer Vision*, 29(2):107–131, Auguest 1998.
13. M. Kass, A. Witkin, and D. Terzopoulos. Snake: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331, 1987.
14. R. Kollarits, C. Woodworth, J. Ribera, and R. Gitlin. An Eye-Contact Camera/Display System for Videophone Applications Using a Conventional Direct-View LCD. *SID Digest*, 1995.

Microsoft Exhibit 1021 - Page 15
Microsoft Corporation v. RealD Spark, LLC
IPR2023-01046 - U.S. Patent No. 10,740,985

15. J. Liu, I. Beldie, and M. Wopking. A Computational Approach to Establish Eye-contact in Videocommunication. In *the International Workshop on Stereoscopic and Three Dimensional Imaging (IWS3DI)*, pages 229–234, Santorini, Greece, 1995.
16. Z. Liu, Z. Zhang, C. Jacobs, and M. Cohen. Rapid Modeling of Animated Faces From Video. *Journal of Visualization and Compute Animation*, 12(4):227–240, 2001.
17. C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. In *IEEE Conf. Computer Vision and Pattern Recognition*, volume I, pages 125–131, June 1999.
18. L. Mhlbach, B. Kellner, A. Prussog, and G. Romahn. The Importance of Eye Contact in a Videotelephone Service. In *11th Interational Symposium on Human Factors in Telecommunications*, Cesson Sevigne, France, 1985.
19. M. Ott, J. Lewis, and I. Cox. Teleconferencing Eye Contact Using a Virtual Camera. In *INTERCHI '93*, pages 119 – 110, 1993.
20. S. Pollard, J. Porrill, J. Mayhew, and J. Frisby. Disparity Gradient, Lipschitz Continuity, and Computing Binocular Correspondance. In O.D. Faugeras and G. Giralt, editors, *Robotics Research: The Third International Symposium*, volume 30, pages 19–26. MIT Press, 1986.
21. S.M. Seitz and C.R. Dyer. View Morphing. In *SIGGRAPH 96 Conference Proceedings*, volume 30 of *Annual Conference Series*, pages 21–30, New Orleans, Louisiana, 1996. ACM SIGGRAPH, Addison Wesley.
22. J. Shi and C. Tomasi. Good Features to Track. In *the IEEE Conferecne on Computer Vision and Pattern Recognition*, pages 593–600, Washington, June 1994.
23. R.R. Stokes. Human Factors and Appearance Design Considerations of the Mod II PICTUREPHONE Station Set. *IEEE Trans. on Communication Technology*, COM-17(2), April 1969.
24. Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.
25. Z. Zhang and Y. Shan. A Progressive Scheme for Stereo Matching. In M. Pollefeys et al., editor, *Springer LNCS 2018: 3D Structure from Images - SMILE 2000*, pages 68–85. Springer-Verlag, 2001.