Toward Object Discovery and Modeling via 3-D Scene Comparison

Evan Herbst Peter Henry Xiaofeng Ren Dieter Fox

Abstract— **The performance of indoor robots that stay in a single environment can be enhanced by gathering detailed knowledge of objects that frequently occur in that environment. We use an inexpensive sensor providing dense color and depth, and fuse information from multiple sensing modalities to detect changes between two 3-D maps. We adapt a recent SLAM technique to align maps. A probabilistic model of sensor readings lets us reason about movement of surfaces. Our method handles arbitrary shapes and motions, and is robust to lack of texture. We demonstrate the ability to find whole objects in complex scenes by regularizing over surface patches.**

I. Introduction

Suppose a mobile robot with an always-on camera makes multiple visits to a location, continuously running SLAM. There is information in the difference between two maps made at different times. We use the PrimeSensor, an *RGB-D* camera that provides dense registered color and depth measurements, to detect objects by means of their movement in 3-D between two maps of the same location.

At a high level this problem is similar to laser scan registration, for which van de Ven et al. [19] have shown it is possible to jointly infer clustering within scans and association between scans. They point out that the efficiency of inference depends heavily on the number of elements to be clustered and associated. Therefore we need to select a small number of surfaces by combining surface elements and, prior to inference, culling those least likely to be part of movable objects. In this work we call this process *object discovery*.

Object discovery in 2-D has been studied by both the computer vision and the robotics communities. In vision, motion segmentation has been used for object discovery assuming highly visually textured surfaces. Bhat et al. [1] assume each surface has enough SIFT matches that RANSAC will find a model for it. That isn't the case for us, partly because our objects occupy relatively little of each image; also we wish to handle textureless objects.

Scene segmentation without motion is difficult, and most work has made restrictive assumptions. Biswas et al. estimate object shape models and object-cluster memberships for pre-segmented object instances in 2-D occupancy grid maps [3]. They assume the number of objects is known and that each connected component of occupied cells is one object. Wolf and Sukhatme perform SLAM for long periods of time by maintaining two occupancy grids, one for static and one for movable objects, to identify cells occupied by movable objects at any given time [20]. Connected components is also the standard approach to segmenting objects in 3-D; for example, [16] assumes all objects are on a table, fits a plane to the table and takes each large remaining connected component of 3-D points to be an object. Several authors have segmented objects by fitting geometric solids to point clouds using the method of [17]. This method was developed to approximate a very dense cloud with a small number of models; its applicability to object modeling is less obvious, since a shape that can be fit well by a moderate number of geometric solids is generally better explained by a single higher-level model, e.g. a train engine. Since we are discovering objects, we don't have a database of predefined object models to fit, so we use only relatively low-level cues. We improve on previous segmentation efforts by identifying surface patches likely to have moved between two scenes. We use ray casting with a probabilistic model of sensor measurements that is robust to sensor noise and that enables us to fuse information from multiple sensor modalities.

Probabilistic measurement models are common in robot localization [7] and mapping [18]. This is natural: generative models are appealing for work with sensors because often we can accurately describe the process of measurement generation, and Bayesian models appeal when we need to combine many sensor readings. Hähnel et al. add dynamic object estimation to SLAM and use expectation-maximization to label each measurement static or dynamic [9]. Kim et al. probabilistically fuse information from multiple depth sensors and cameras for dense 3-D reconstruction [13]. We use a generative sensor measurement model with similar sensor modalities to those of [13], but discard some of their independence assumptions.

Probabilistic sensor models have also been used for change detection. Kaestner et al. [12] use a model of laser measurements to find changes in outdoor scenes. They perform global alignment of laser scans using iterated closest points (ICP) and detect changes using statistical significance testing for each measurement independently.

Our algorithm aligns the frames of each RGB-D video using SLAM, then reconstructs each scene as a dense set of surface elements (Section II). Our measurement model (Section III) tells us how likely each surface element is to have moved between two scenes. In Section IV-A we employ the model to detect differences between scenes and in Section IV-B we spatially regularize to find large surface regions that may belong to movable objects.

E. Herbst, P. Henry and D. Fox are with the University of Washington, Department of Computer Science & Engineering, Seattle, WA 98195. X. Ren and D. Fox are with Intel Labs Seattle, Seattle, WA 98105.

This work was funded in part by an Intel grant, by ONR MURI grants N00014-07-1-0749 and N00014-09-1-1052, by the NSF under contract IIS-0812671, and through the Robotics Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement W911NF-10-2-0016.

(a) (b) (c)

Fig. 1: (a), (b) reconstruction of two scenes; (c) their superimposition after alignment.

(a) (b)

Fig. 2: Zoom into the edge of the table from Fig. 1: (a) after SIFT RANSAC alignment; (b) after additional ICP alignment.

II. Scene Reconstruction and Alignment

The inputs to our system are two or more RGB-D videos representing separate visits to the same location. From each sequence we generate a geometrically consistent 3-D reconstruction, or *scene*. Objects can move between visits but are assumed static during each visit.

A. Scene Reconstruction / SLAM

We build on RGB-D Mapping [10], a SLAM technique for generating dense 3-D maps. At a high level, RGB-D Mapping performs (1) pairwise frame alignment by visual odometry, (2) loop-closure detection and (3) global path optimization. In our off-line setting, we first compute visual features for all video frames. We choose SIFT as the descriptor, because accuracy is more important than speed to our application. Each visual feature is assigned the depth of the nearest pixel. The 3-D transformation between each pair of adjacent frames is estimated using RANSAC, using Horn's method [11] to fit models.

Loop-closure detection selects a subset of frames to be *keyframes* and speeds up the search by only matching pairs of keyframes. The same RANSAC algorithm as before is used to estimate the 3-D transformation between matching keyframes. Each time-adjacent frame pair and each pair of matched keyframes generates an edge in a *pose graph*, described by the transform produced by RANSAC. We use a pose-graph optimizer to globally adjust camera poses. The constraints used by this optimizer do not encode all the information from feature matching, so the optimization may introduce local inconsistencies. Therefore, we improve on RGB-D Mapping by following pose-graph optimization with bundle adjustment, again using 3-D-augmented SIFT descriptors. Bundle adjustment improves local consistency without disrupting global consistency.

Each frame consists of 250,000 colored 3-D points. Representing a scene by the concatenation of these point clouds is redundant, unwieldy and noisy. Following global registration, we convert points to *surfels* [14] to reduce map size and smooth evidence from multiple frames. Each 3-D point from each frame is assigned to a surfel, either an existing one or a newly created one if there is no existing surfel close to the point. Fig. 1 shows two reconstructions of a location.

We probabilistically model the position, color and orientation of each surfel, keeping uncertainty information as well as MAP values. A Gaussian over position can be updated recursively during reconstruction, and we compute distributions over color and normal after reconstruction. We exclude from differencing each surfel that has high uncertainty in any of these attributes.

The rest of our algorithm will require the surface to be representable as a finite set of local samples. We choose to use surfels for this paper, but the algorithm would work equally well with meshes or samples from implicit surfaces.

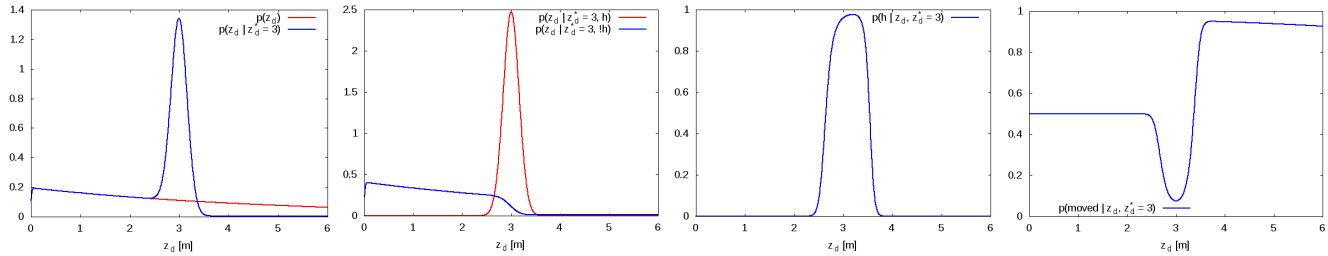
B. Scene Alignment

Assuming moved objects are a small part of each scene, we fit a single rigid transformation to the sets of point features in each pair of scenes using RANSAC. We compute SIFT features, assign each the 3-D location of the nearest depth point in its frame, and transform each image's features into a scene-global frame. This allows us to compute a transform between the sets of all features in each scene, which is more robust than the alternative of computing transformations between individual frames of different scenes and picking one to be the global transform. These transforms are inaccurate due to depth noise, so we furthermore run ICP, using the point-to-plane error of [5]. We ignore 30% of the worst-aligned points when calculating error, to account for the movable objects we expect. As an example of our global alignments, Fig. 1 shows the superimposition of two aligned scenes. A typical example of the improvement achieved by ICP is shown in Fig. 2.

III. Measurement Model for RGB-D Cameras

We now introduce a sensor model for RGB-D cameras. Our model incorporates information from both depth and color, taking the correlations between the different sensor modalities into account.

RGB-D cameras provide a dense matrix of pixels, each measuring the color and distance of the closest surface in the pixel direction [10]. We make use of dense depth information to locally estimate surface normals and define a virtual



(a) (b) (c) (d)

Fig. 3: (a) Probability of a depth measurement z_d given no knowledge about the environment (red) and given a model in which the closest surface patch along the pixel beam is at 3 m (blue). (b) Probability of measuring distance z_d given that the known surface caused (red) or did not cause (blue) the measurement. (c) Probability that the surface at $z_d^* = 3$ caused the measurement z_d. This probability serves as a weighting function for the mixture components of the color and orientation models. (d) Probability that the expected surface is missing given a depth measurement z_d. The model parameters used to make these plots are different from those used in our experiments.

measurement $z = \langle z_d, z_c, z_\alpha \rangle$, where z_d is the depth value of the pixel, z_c the color, and z_α the orientation of the detected surface. Our probabilistic model for pixel measurements is motivated by beam-based models for distance sensors such as laser range finders or sonar sensors, extending these to incorporate the additional color and local shape information provided by RGB-D cameras.

As will become clear in Section IV, we need to consider two cases, one in which there is no knowledge about the environment and one in which we have a 3-D reconstruction of the environment we expect to see.

A. No Scene / Surface Information

We now model an RGB-D pixel measurement z if there is no prior information about surfaces in the scene. We assume that in absence of any surface information, the depth, color, and orientation measurement components are independent:

$$p(z) = p(z_d, z_c, z_\alpha) \;\approx\; p(z_d)p(z_c)p(z_\alpha) \tag{1}$$

Following models developed for range sensors such as laser scanners and sonar sensors [18], [8], the distribution over depth measurements given no information about surfaces is a mixture between an exponential and a uniform distribution:

$$\begin{aligned} p(z_d) &= \begin{pmatrix} w_{\text{short}} \\ w_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} p_{\text{short}}(z_d) \\ p_{\text{rand}}(z_d) \end{pmatrix} \\ p_{\text{short}}(z_d) &= \begin{cases} \eta\, \lambda_{\text{short}}\, e^{-\lambda_{\text{short}} z_d} & \text{if } z_d \le z_{\max} \\ 0 & \text{otherwise} \end{cases} \\ p_{\text{rand}}(z_d) &= \begin{cases} \frac{1}{z_{\max}} & \text{if } z_d < z_{\max} \\ 0 & \text{otherwise} \end{cases} , \end{aligned} \tag{2}$$

where p_{short} models the probability that the pixel detects a surface before reaching maximum range, and p_{rand} is a uniform distribution over the measurement range. w_{short} and w_{rand} are mixture weights and λ_{short} and η are the scale parameter of the exponential distribution and its normalizer, respectively. An example for this mixture distribution is represented by the red line in Fig. 3(a).

Given no map of the environment, we model the color and orientation distributions $p(z_c)$ and $p(z_\alpha)$ as uniform distributions over the color and orientation spaces.

B. Known Scene / Surface

We now assume that a scene surface map is available. Given the location of the depth camera, we can use ray tracing to determine the distance, color, and local orientation of the closest surface along the pixel direction. This information can be encoded in an *expected* measurement $z^* = \langle z_d^*, z_c^*, z_\alpha^* \rangle$. Conditioning on the expected measurement and separating the individual components of the pixel measurement, we apply the chain rule:

$$\begin{aligned} p(z \mid z^*) &= p(z_d, z_c, z_\alpha \mid z^*) \\ &= p(z_d \mid z^*)p(z_c \mid z_d, z^*)p(z_\alpha \mid z_d, z_c, z^*) \\ &\approx p(z_d \mid z_d^*)p(z_c \mid z_d, z^*)p(z_\alpha \mid z_d, z^*) \,. \end{aligned} \tag{3}$$

In the last step we assume that the orientation z_α is independent of the color z_c. This model is similar to that of [13], which combines three cues—depth, color, and image gradients—for 3-D reconstruction. However, while they assume independence of the components, we explicitly model the dependences between color, orientation, and depth. As we will show below, this results in more accurate models since, given a surface patch, the distance provides additional information about the expected orientation and color.

We now derive models for the three components. The model for the depth value $p(z_d \mid z_d^*)$ is almost identical to that of existing models for laser or sonar beams. Here, we stay close to the one given by Thrun and colleagues [18], who model $p(z_d \mid z_d^*)$ as a mixture of four distributions [1]. We use three of the components [2]:

$$p(z_d \mid z_d^*) = \begin{pmatrix} w_{\text{hit}} \\ w_{\text{short}} \\ w_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} p_{\text{hit}}(z_d \mid z_d^*) \\ p_{\text{short}}(z_d \mid z_d^*) \\ p_{\text{rand}}(z_d \mid z_d^*) \end{pmatrix} \tag{4}$$

p_{hit} represents the case in which the measurement hits the expected surface, and the other two components are just like those modeling unknown surfaces, with the exponential distribution cut off at the expected distance. The distribution

[1] We actually use the more correct but less intuitive model of [7].

[2] We do not explicitly model maximum-range measurements, since these correspond to uninformative noise in our RGB-D camera and can be ignored without substantial loss of expressiveness. Incorporating max range if necessary would be straightforward.

for the first case is a Gaussian centered at the expected distance:

$$p_{\text{hit}}(z_d \mid z_d^*) = \begin{cases} \eta\, \mathcal{N}(z_d; z_d^*, \sigma_{\text{hit}}^2) & \text{if } z_d \leq z_{\max} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Here $z_{\max}$ is the maximum measurement range and η is a normalizer. σ_{hit}^2 is the distance measurement noise, which is a function of z_d^*, of the obliqueness of the viewing angle to the expected surface, and of the error in stereo depth estimation as a function of depth (we use a stereo noise model since our depth camera technology is based on stereo). Intuitively, the more obliquely the camera views a surface, the less accurate the measurement of that surface will be; we model this by

$$\sigma_{\text{hit}} = \frac{\sigma_0(z_d^*)}{\sin(\pi/2 - \theta)},$$

where θ is the angle between the viewing direction and the expected surface's normal. An example for the mixture distribution is given by the blue line in Fig. 3(a). Fig. 3(d) visualizes the posterior $p(\text{m} \mid z_d, z_d^*)$ resulting from ignoring color and orientation.

We now turn to the models for the color and orientation components of a measurement, respectively $p(z_c \mid z_d, z^*)$ and $p(z_\alpha \mid z_d, z^*)$. The color model obviously should depend on z_c^*, the color of the expected surface. However, the additional conditioning on z_d allows us to use the depth value to determine whether the measurement was actually caused by the expected surface or by some other surface (due to moving objects, for example). To do so, we introduce a binary random variable h whose value is whether the expected surface caused the measurement.

$$\begin{aligned} p(z_c \mid z_d, z^*) = {} & p(z_c \mid z_d, z^*, h)\, p(h \mid z_d, z^*) \\ & + p(z_c \mid z_d, z^*, \neg h)\, p(\neg h \mid z_d, z^*) \end{aligned} \quad (6)$$

follows directly by conditioning on h. We can model the first component of this mixture by a Gaussian with mean at the expected color:

$$p(z_c \mid z_d, z^*, h) \sim \mathcal{N}(z_c^*, \sigma_{\text{col}}^2) \quad (7)$$

We model the other mixture component in (6), $p(z_c \mid z_d, z^*, \neg h)$, by a uniform distribution over the color space.

To determine the weights of these two mixture components, we apply Bayes' rule to $p(h \mid z_d, z^*)$ and $p(\neg h \mid z_d, z^*)$, yielding

$$p(h \mid z_d, z^*) \propto p(z_d \mid h, z^*) p(h \mid z^*) \quad (8)$$

$$p(\neg h \mid z_d, z^*) \propto p(z_d \mid \neg h, z^*) p(\neg h \mid z^*) \quad (9)$$

with the same constant of proportionality in both. Here, $p(z_d \mid h, z^*)$ and $p(z_d \mid \neg h, z^*)$ are special cases of the distance model given in (4): $p(z_d \mid h, z^*)$ is the Gaussian component p_{hit}, and $p(z_d \mid \neg h, z^*)$ is a mixture of the other two components p_{short} and p_{rand}.

$p(h \mid z^*)$ in (8) can be computed as $\int_0^{z_{\max}} \frac{p_{\text{hit}}(z_d \mid z^*)}{p(z_d \mid z^*)} dz_d$. Example distributions for $p(z_d \mid h, z^*)$ and $p(z_d \mid \neg h, z^*)$ are given in Fig. 3(b). Plugging these distributions into the weighting function (8) results in the plot shown in Fig. 3(c). Again, this distribution serves as a weighting function for the mixture components in the color model (6). As a result, the measured color z_c should be similar to the expected color z_c^* if the depth measurement z_d is close to the surface distance z_d^*, and uniformly distributed otherwise.

The derivation of the orientation model $p(z_\alpha \mid z_d, z^*)$ is analogous and omitted for brevity. We model orientation by a von Mises-Fisher distribution, a common choice of distribution over a hypersphere because it has a closed-form pdf [2]. We use parameters $\mu = z_\alpha^*$ and $\kappa = 60$.

One way to balance evidence from these three components is to change the effective number of observations [18]. We use the reweighting

$$p(z \mid z^*) = p(z_d \mid z_d^*)^\alpha p(z_c \mid z_d, z^*)^\beta p(z_\alpha \mid z_d, z^*)^\gamma. \quad (10)$$

For all experiments below we use $\alpha = 1, \beta = 25, \gamma = 5$ to allow the color and normal components to override the depth component when the observed and expected depths are similar. These numbers are an educated guess, and results are robust to a wide range of values.

IV. Scene Differencing

A. Pointwise Motion Estimation

We are now equipped to detect differences between two 3-D maps. Consider a single surface patch in the first scene. Patch $s = \langle s_x, s_c, s_\alpha \rangle$ is described by its location s_x, color s_c, and surface orientation s_α. Using the alignment technique described in Section II, we can determine all camera poses of the second scene relative to s. Based on these camera poses we identify for each camera frame the pixel that points at that surface patch (if the patch is in the camera's field of view). From the camera pose and the surface descriptor s we can also compute the expected measurement z^*. We wish to determine for each s the probability that it *moved* away from its location in the first scene.

Let $\mathbf{z}_s$ denote the set of measurements taken in the second scene associated with a specific surface patch s in the first scene, and let $\mathbf{z}^*{}_s$ denote the expected measurements computed from $\mathbf{z}_s$ and s. We denote by m the Boolean variable representing whether s moved. By Bayes' rule,

$$p(\text{m} \mid \mathbf{z}_s, \mathbf{z}^*{}_s) = \frac{p(\text{m}, \mathbf{z}^*{}_s) p(\mathbf{z}_s \mid \text{m}, \mathbf{z}^*{}_s)}{p(\mathbf{z}_s, \mathbf{z}^*{}_s)} \quad (11)$$

$$\approx \frac{p(\text{m}) p(\mathbf{z}^*{}_s) p(\mathbf{z}_s \mid \text{m}, \mathbf{z}^*{}_s)}{p(\mathbf{z}_s \mid \mathbf{z}^*{}_s) p(\mathbf{z}^*{}_s)} \quad (12)$$

$$\propto p(\text{m}) \prod_{i=1}^{I} p(z_i \mid \text{m}, z_i^*)\,, \quad (13)$$

where I is the number of frames in the second scene, $p(\text{m})$ comes from prior knowledge (we usually use .1 because few of our surfels move) and

$$p(z_i \mid \text{m}, z_i^*) = \begin{cases} p(z_i \mid z_i^*), & \text{m} = 0 \\ p(z_i), & \text{m} = 1 \end{cases}. \quad (14)$$

The idea behind (14) is that if the surface did not move, the measurement follows the surface-based model (3). If the surface patch moved, however, we have no knowledge about what to expect, so we use the prior model (1).

The pointwise motion probabilities computed for the surface patches of the left scene in Fig. 1 are shown in Fig. 5(a). The changed objects are detected as high-probability areas, using the combination of depth, color, and orientation cues. Furthermore, the grey areas appropriately represent surface regions that are occluded in the second scene (Fig. 1(b)). However, due to sensor noise and imperfect alignment, the individual motion probabilities are still noisy.

B. Spatial Regularization

To improve spatial consistency we generate a Markov random field with a node for each surface patch. Each node has two choices of label: moved and not moved. For efficiency we use only pairwise cliques in the MRF so that we can use graph cuts [4] for inference. The data energy for the MRF is straightforwardly calculated from the output of the pointwise model:

$$E_d(l) = \begin{cases} \log(1 - p(\mathrm{m} \mid \mathbf{z}, \mathbf{z}^*)), & l = \mathrm{m} \\ \log(p(\mathrm{m} \mid \mathbf{z}, \mathbf{z}^*)), & l = \neg\mathrm{m} \end{cases} \tag{15}$$

For the smoothness energy we use the Potts model, weighted by the curvature of the local region to discourage cuts in low-curvature areas:

$$E_b(s_i, s_j, l_1, l_2) = w_s \begin{cases} \frac{1}{\sqrt{\max(\kappa_i, \kappa_j, \kappa_0)}}, & l_1 \neq l_2 \\ 0, & l_1 = l_2 \end{cases} \tag{16}$$

Here s_i and s_j are surface patches and l_1 and l_2 their labels. κ_i is the previously computed local curvature at s_i. κ_0 is a constant representing the minimum curvature beyond which all surfaces can be described as "very flat"; we set it to 1 m^{-1}. We set the smoothness weight w_s by optimization over two scene pairs (not the ones visualized in Fig. 5 and Fig. 6). The result of applying this MRF to the probabilities displayed in Fig. 5(a) is shown in Fig. 5(b).

V. Experiments

We evaluate our sensor model and demonstrate the ability to discover and model objects in a tabletop setting. In each experiment, the RGB-D camera was carried by hand to generate partial coverage of the scene.

A. Depth-Dependent Color Model

Our color and orientation models take the expected color and orientation into account, but also model dependence on the expected vs. measured distance of the surface patch. In this experiment we demonstrate the advantage of our model over a sensor model that ignores this dependence (as that of [13]). The left two panels in Fig. 4 show top views of an experiment in which a white object is placed on the table in scene 1 (Fig. 4(a)) and is then occluded from the camera viewpoints by a white box in scene 2 (Fig. 4(b)). Fig. 4(c) shows the motion probabilities coming from a color model that does not model dependence on depth (a depth-independent mixture between a Gaussian and uniform color model). The model erroneously is certain that the white object in the left scene did *not* move (note the nearly black outline of the object). However, the second scene provides no evidence about this area since it is fully occluded by the white box. As can be seen in Fig. 4(d), our depth-dependent color model generates the correct probabilities, having near 0.5 probability in the occluded area.

B. Detecting Moved Objects

In Fig. 5 we show results on the complex scene pair of Fig. 1. Each scene contains a spray bottle and a cylindrical object in about the same location, such that simple nearest-neighbor point search would consider them the same objects. Due to our use of color and orientation information, we correctly detect changed surfaces at these locations in both scenes. One of the objects is also occluded, necessitating the use of our depth-dependent color model. In Fig. 5(a) and Fig. 5(c) we show the raw probabilities $p(\mathrm{m} \mid z_d, z^*)$ from our sensor model. Labels after regularization are shown in Fig. 5(b) for the first scene and Fig. 5(d) for the second scene, demonstrating that we find all movable objects with high accuracy despite using only low-level cues. We extract each connected component of same-label surface patches as a 3-D model and show the foreground (object) models in Fig. 7(a), along with the merge of the two background segments, which fills in some holes in each individual table.

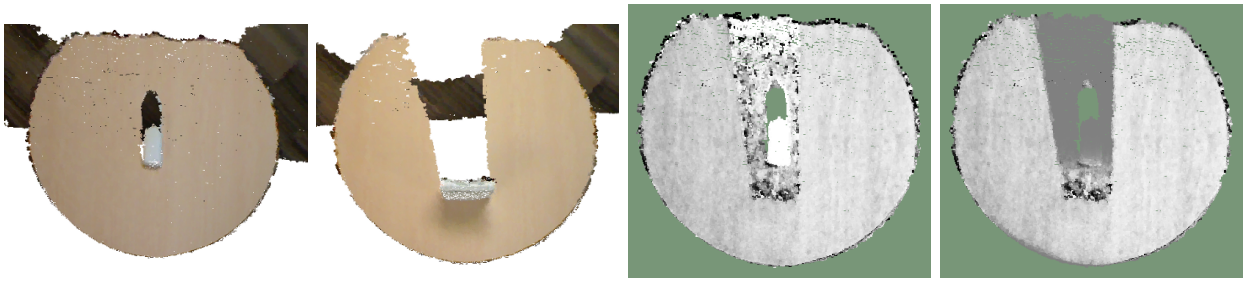
Fig. 6 gives results for another scene pair. In the tvtable dataset, small objects move around on a cluttered table. In Fig. 7(b) we show four extracted foreground connected components along with the merged backgrounds; again we have filled in some holes on the table. We do not show foreground components that are very small, of which there are one in Fig. 5 and three in Fig. 6. We provide 3-D model files for these scenes, and the extracted objects, at `http://www.cs.washington.edu/robotics/projects/object-discovery/`. PLY files can be opened using Meshlab [6].

C. Quantitative Results

We manually annotated which surfels moved for each scene pair in a set of four scenes. (Occluded points were marked not moved.) Performance numbers for these scene pairs are given in table I. Precision and recall were calculated with respect to surfels annotated as moved, since those are a small minority of surfels and are the most salient surfels for most applications of differencing.

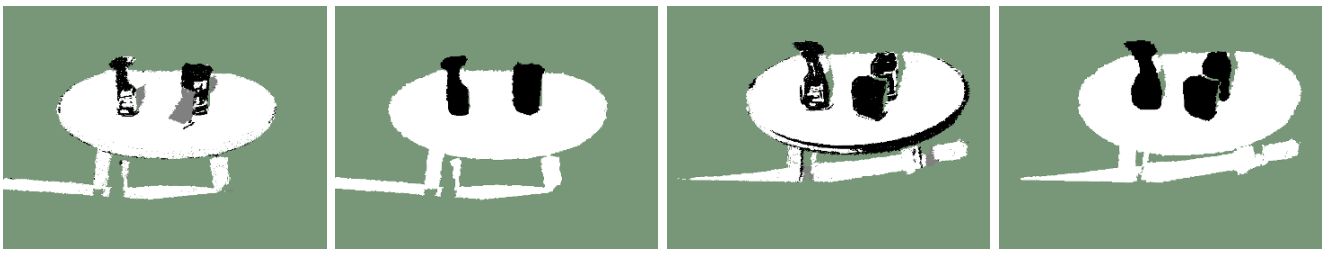
	precision	**recall**	**accuracy**	**baseline accuracy**	**% error reduction**
average	.965	.800	.990	.968	68.8
min	.911	.662	.980	.954	

TABLE I: Performance statistics aggregated over 12 scene pairs after spatial regularization. "Baseline" refers to the classifier that labels all surfels not moved. "Error reduction" is our improvement w.r.t. the baseline error rate. To pick regularization parameters we optimized a modified F-1 score giving extra importance to precision.



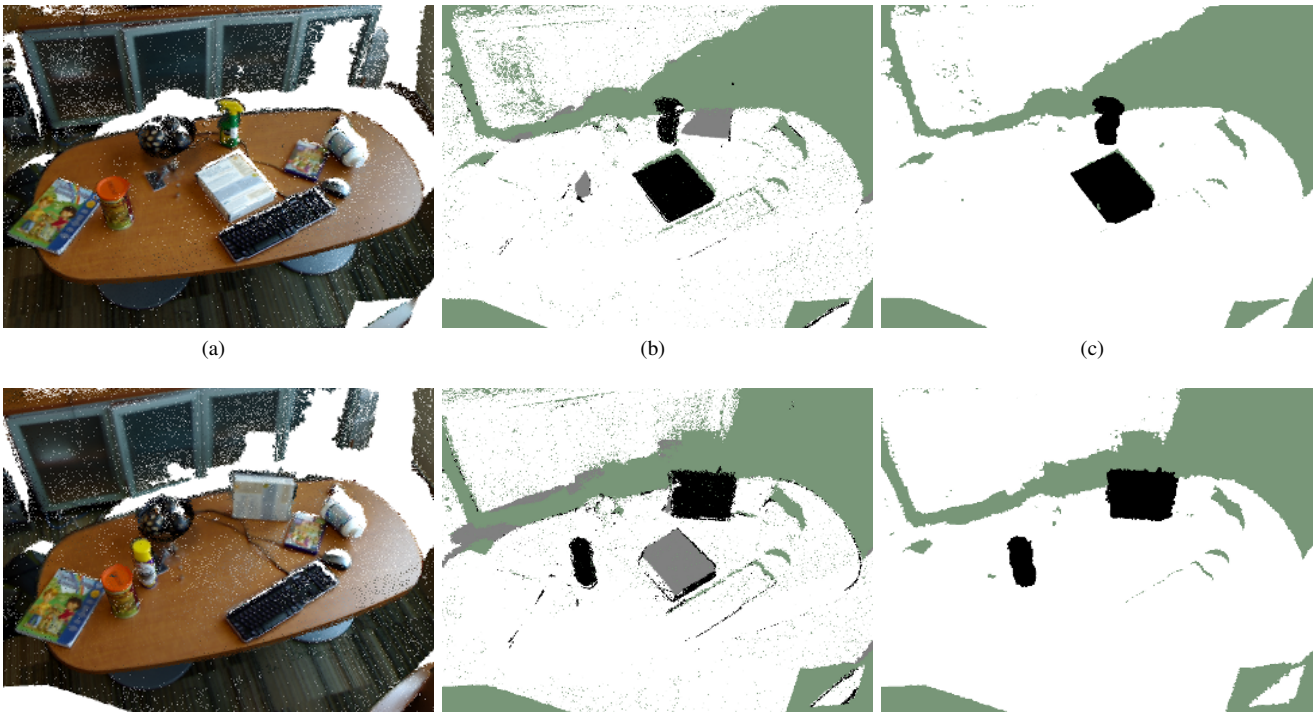
(a) (b) (c) (d)

Fig. 4: Depth-dependent color model. The white object in the left scene (a) is occluded by the white box in the second scene (b). A depth-independent model (c) generates a very high probability that the white object did *not* move. Our color model (d) correctly captures the occlusion and generates motion probabilities near 0.5 in the occluded area. In this and further figures, green represents points either not seen during mapping or seen in one map but not the other so that differencing is not possible. In (c) and (d), the floor has also been removed for clarity. Grayscale pixels give $p(\mathrm{m} \mid z)$: the darker, the higher the probability of movement.



(a) (b) (c) (d)

Fig. 5: (a) surfel-wise differencing and (b) regularization results for the first scene of Fig. 1; (c), (d) similarly for the second scene.



(a) (b) (c)

(d) (e) (f)

Fig. 6: (a), (d) two scenes from the tvtable dataset; (b), (e) surfel-wise differencing results for the two scenes; (c), (f) regularization results. A few surfels shown in the surfelwise result do not appear in the final output due to culling we perform prior to regularization. Three large occluded areas in (b) and (e) show as medium gray (movedness uncertain).

D. Timing

In table II we break down the time spent by our system on the scenes of Fig. 6 (differencing both ways). These two videos have 360 and 500 frames respectively; the maps contain 650k and 620k surfels respectively. Subsampling frames would speed up mapping, reconstruction and differencing.

Fig. 7: Objects extracted from the scene pairs of Fig. 1 and Fig. 6, and the merged background components of each pair of scenes. In the right panel some of the background is faded for clarity.

There are a number of other ways to speed up mapping, but in our experience they are all very data-dependent; we used settings conservative enough to work in a relatively large number of cases at the expense of speed. Alignment could be sped up by subsampling the set of features given to RANSAC; here again we are conservative. The machine used has 16 hardware threads, and various parts of our system make use of this; in particular differencing is almost embarrassingly parallel.

Stage	Time
preprocessing	400s
mapping	600s (mostly bundle adjustment)
reconstruction	360s
alignment	140s (100s w/o ICP)
differencing	180s
regularization	24s

TABLE II: Time spent in each stage of our algorithm for the scene pair shown in Fig. 6.

VI. Conclusion

We have described a method to determine movable parts of a scene based on differencing with another scene. Our method makes no assumption on overall object shape, texture of surfaces, or large motion relative to object size. It also models and finds occluded surfaces. Phrasing the differencing problem in terms of a probabilistic model of sensor readings provides robustness to noisy data as well as a principled way to combine information from multiple sensors. The method relies on 3-D reconstruction, whose accuracy and robustness could use improvement; the difficulty of making good geometric maps is our largest obstacle. Our algorithm can also find large moving objects, such as furniture; see the website (given in Section V-B) for an example.

We plan to use the movement detection techniques presented here together with matching of surfaces among multiple scenes to improve object localization. With our differencing technique it is straightforward to combine evidence from multiple scenes to improve the input to spatial regularization. Matching can be initialized using point-feature matches for highly textured areas and point-cloud-based techniques for untextured regions. Then evidence about intra-scene and inter-scene similarity can be combined, perhaps using random fields similar to those of [19], to identify objects common to multiple scenes.

References

[1] P. Bhat, K. Zheng, N. Snavely, A. Agarwala, M. Agrawala, M. Cohen, and B. Curless. Piecewise image registration in the presence of large motions. In *CVPR*, 2006.

[2] C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[3] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in dynamic environments with mobile robots. In *IROS*, 2002.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 1999.

[5] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image Vision Computing*, 10(3):145–155, 1992.

[6] P. Cignoni. Meshlab. `http://meshlab.sourceforge.net/`.

[7] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments, 1999.

[8] D. Fox, W. Burgard, and S. Thrun. Markov localization for reliable robot navigation and people detection. In *Modelling and Planning for Sensor-Based Intelligent Robot Systems*, LNCS. Springer, 1999.

[9] D. Haehnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *ICRA*, 2003.

[10] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3-D modeling of indoor environments. In *ISER*, 2010.

[11] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society A*, 1987.

[12] R. Kaestner, S. Thrun, M. Montemerlo, and M. Whalley. A non-rigid approach to scan alignment and change detection using range sensor data. In *Symposium on Field and Service Robotics*, 2005.

[13] Y. Kim, C. Theobalt, J. Diebel, J. Kosecka, B. Miscusik, and S. Thrun. Multi-view image and TOF sensor fusion for dense 3-D reconstruction. In *3DIM*, 2009.

[14] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *SIGGRAPH*, 2000.

[15] M. Ruhnke, B. Steder, G. Grisetti, and W. Burgard. Unsupervised learning of 3-D object models from partial views. In *ICRA*, 2009.

[16] R. Rusu, N. Blodow, Z. Marton, and M. Beetz. Close-range scene segmentation and reconstruction of 3-D point cloud maps for mobile manipulation in domestic environments. In *IROS*, 2009.

[17] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 2007.

[18] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[19] J. van de Ven, F. Ramos, and G. Tipaldi. An integrated probabilistic model for scan matching, moving object detection and motion estimation. In *ICRA*, 2010.

[20] D. Wolf and G. Sukhatme. Online simultaneous localization and mapping in dynamic environments. In *ICRA*, 2004.