2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Toward Online 3-D Object Segmentation and Mapping

Evan Herbst Peter Henry Dieter Fox

Abstract— **We build on recent fast and accurate 3-D reconstruction techniques to segment objects during scene reconstruction. We take object outline information from change detection to build 3-D models of rigid objects and represent the scene as static and dynamic components. Object models are updated online during mapping, and can integrate segmentation information from sources other than change detection.**

I. Introduction

Service robotics is an expanding field. Indoor service robots need to understand their environments geometrically, topologically, and in terms of objects and other semantic concepts. Accurate 3-D object models are used in training object detectors and in most approaches to object manipulation. This paper makes progress toward acquiring detailed 3-D object models with as little human intervention as possible.

Over the last few years robotic vision systems have gotten much better at detailed and very fast 3-D reconstruction of static scenes using volumetric approaches running on the GPU [1], [2], [3]. The ability to detect scene changes between detailed 3-D maps was demonstrated in [4]. In this paper we demonstrate high-quality object segmentation and modeling *during* scene reconstruction.

Imagine that a robot makes recordings of the same scene at several points in time, separated by perhaps a day—long enough for some objects to have moved. We should be able to partition the 3-D reconstruction of the scene into a "background map" of everything that never moves and a set of dynamic rigid objects that move at least once betwen two mapping visits. We use change detection to provide segmentation information, because over long periods of time, objects that can move probably will be moved.

One way in which object segmentation with change detection can fail is in being unable to separate objects that move to or from a spatially adjacent configuration. Another issue is that a robot seeing a previously unobserved region of space has no change detection information for the region. To address these problems, our mapping framework is able to take in information about segmentation in the current scene that does not come from change detection. We demonstrate this capability by using information from a human-provided movement of a single object that can improve our segmentation of that object and others that it touches. This is complementary to work (e.g. [5]) in which a robot pushes objects to make them move and attempts to segment all moved objects using dense motion estimation. We use some of the same techniques used therein, but as that work demonstrates, segmentation from motion data without very strong regularizers (e.g., a very strong bias for the idea that objects are entirely convex) doesn't currently work very well. Therefore, here we instead acquire known-correct segmentation information from a human teacher.

Our method avoids requiring visual point features, except during background alignment (see sec. III). One of our goals is to as much as possible build a principled framework around one main tool, change detection, used in a variety of ways, to combat the tendency of mobile robotics systems toward heuristics and magic numbers. All of our modeling happens online.

This paper is organized as follows. In sec. II we discuss related work. Sec. III explains our reconstruction and change detection, and sec. IV discusses how we use them in an online fashion. Sec. V introduces our object modeling. We discuss using additional segmentation information in sec. VI. Finally we mention efficiency in sec. VII and future work in sec. VIII.

II. Related Work

We cannot list all of the immense amount of work in object segmentation and modeling with vision; we hope to provide a useful sample.

Representing scenes using objects and background is an old idea. In mobile robotics it goes back at least to Biswas et al. [6], who process a set of previously aligned 2-D occupancy grid maps with change detection in an off-line fashion to identify movable objects. Haehnel et al. [7] separate moving and static regions of 2-D laser scans in an online fashion in order to navigate through moving obstacles. They also model moving objects, by extending the 2-D profiles of dynamic regions through a 3-D laser scan. Finman et al. [8] use a combination of 3-D change detection (using depth but not color) and running per-object learned segmentation models on each of a set of maps to extract a set of potential objects that could be input to an object discovery algorithm. Their algorithm is also off-line, but runs on multiple large-scale maps built separately. Probably the most similar work to this paper is that of Alimi et al. [10], who create an RGB-D background map after running change detection on many pairs of temporally adjacent views of a scene. Their camera is static and their views are single frames. There is also work on linking maps of objects and background using semantic relations: in [11], Wurm et al. analyze a laser scan to identify objects and background, using the table-plane assumption and other heuristics to avoid solving segmentation. They create octree maps of objects

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

This work was funded in part by an Intel grant, the NSF under contract IIS-1227234, and ONR MURI grant N00014-07-1-0749.

and background separately, potentially at different spatial resolutions, and connect them with semantic concepts such as "supported-by". Salas-Moreno et al. [9] match the scene to previously built geometric object models at each frame during reconstruction in order to build a separate information layer with object poses and labels. They update the poses of matched objects online as the 3-D reconstruction changes.

Online motion segmentation is also an active area of research. Pundlik and Birchfield [12] match point features in RGB-D video as input to sparse motion segmentation, and use spatial regularization over pixels to extend the segmentation to a dense one. Roussos et al. [13] perform motion estimation and reconstruction of 3-D rigid objects from RGB video in an off-line way, using recent optimization techniques to allow them to include all pixels in the optimization objective.

This paper improves on previous work that segments objects using change detection given videos of multiple visits to a scene, assuming the scene is static during each visit. We can use static SLAM to build a static map from each video. Change detection run over pairs of these maps gives us some information about the geometry of movable objects in the scene. We can merge the unchanged parts of each map into an overall "background map"; the other segments from all scenes are models of moved objects. This approach was taken in [14], [8]. In this work, we instead perform change detection in an online fashion, comparing each new video frame to a previous map *while running SLAM* instead of mapping each scene before detecting changes. There are three major advantages to the online approach. Firstly, Newcombe et al. [1] demonstrate improved alignment performance during SLAM from aligning new frames to existing maps rather than aligning frames to each other in traditional visual odometry style. In experiments not detailed herein we have found a similar effect when aligning frames from one video to a map made from another video for change detection purposes. Secondly, if we perform change detection during reconstruction rather than afterward, we can improve the quality of models of objects present in the current scene (the scene that we're processing a frame at a time). Because we know, given partial object views, the approximate sizes of objects, we can make high-resolution models of small regions including these objects, while mapping the rest of the scene in relatively low resolution to save memory and time. Thirdly, we want to extend this work to involve *active segmentation*, in which the robot can move the camera and/or objects being modeled in order to get more informative views. This is only possible if objects are modeled in real time.

The research contributions of this paper are as follows. We combine the idea of making separate maps for static background and dynamic objects with recent advances in online reconstruction. We apply frame-to-map alignment, shown by Newcombe et al. [1] to improve single-map alignment, to alignment between multiple maps. We discuss what information change detection gives us for the purpose of modeling our knowledge about object boundaries.

III. Change Detection

If the robot visits a room, goes away and comes back hours or days later, probably some objects have moved. We can find them by using *change detection*, looking for statistically significant differences between two models of the same phenomenon. In our previous work [4] we use change detection to compare two or more maps produced by a SLAM algorithm. Our main building block is a change detection algorithm operating on one 3-D map and one RGB-D frame. One weakness of that work is that each map is reconstructed fully in order to align the frames of each map to each other map. The global consistency of static SLAM is poor enough that these frame-to-map alignments can be improved. In this work we do so by aligning each frame of a video separately to a map made from previous videos before running change detection. We have found this to improve change detection and dense alignment.

Our change detection procedure takes as input a relative pose between two scenes (in this paper, an aggregated map $\mathcal{M}$ and an RGB-D frame $\mathcal{F}$), renders $\mathcal{M}$ into $\mathcal{F}$'s camera, and compares sensor measurements at each pixel to decide whether the surface appearing at that pixel in $\mathcal{M}$ also appears at that pixel in $\mathcal{F}$. The decision is made mostly on a per-pixel basis, using the "expected" and "observed" attributes rendered from $\mathcal{M}$ and $\mathcal{F}$ respectively. Change detection is asymmetric: it matters which scene is considered the "observation". We use a probabilistic noise model for RGB-D sensors based on that used in our previous work. Let $\mathbf{z}_s =< z_d, z_c, z_\alpha >$ denote the set of measurements (depth, color, surface normal) taken from $\mathcal{F}$ associated with a surface patch s in $\mathcal{M}$, and let $\mathbf{z}^*{}_s =< z_d^*, z_c^*, z_\alpha^* >$ denote the expected measurements computed from s. We denote by m the Boolean variable representing whether s changed. From [4],

$$p(m \mid \mathbf{z}_s, \mathbf{z}^*{}_s) = \frac{p(m, \mathbf{z}^*{}_s) p(\mathbf{z}_s \mid m, \mathbf{z}^*{}_s)}{p(\mathbf{z}_s, \mathbf{z}^*{}_s)} \tag{1}$$

$$\propto p(m) p(\mathbf{z}_s \mid m, \mathbf{z}^*{}_s) \; . \tag{2}$$

We now want two sensor noise models $p(\mathbf{z}_s \mid m = 1, \mathbf{z}^*{}_s)$ and $p(\mathbf{z}_s \mid m = 0, \mathbf{z}^*{}_s)$ to compare at each pixel the hypotheses that the surface at that pixel in one scene has ($m = 1$) and has not ($m = 0$) moved in the other. For this paper we modify the model that assumes the surface has not moved. In this case, we condition the color and normal distributions on the depth distribution. This 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 occluders, for example). To do so, we introduce a binary random variable h whose value represents whether the expected surface caused the measurement.

$$\begin{aligned} p(z \mid m=0, z^*) &= p(z_d, z_c, z_\alpha \mid z^*) && (3)\\ &= p(z_d \mid z_d^*) p(z_c \mid z_d, z^*) p(z_\alpha \mid z_d, z_c, z^*) \\ &\approx p(z_d \mid z_d^*) p(z_c \mid z_d, z^*) \mathbf{p}(\mathbf{z}_\alpha \mid \mathbf{z_d}, \mathbf{z}^*) \\ p(z_c \mid z_d, z^*) &= \mathbf{p}(\mathbf{z_c} \mid \mathbf{z_d}, \mathbf{z}^*, \mathbf{h}) \, p(h \mid z_d, z_d^*) && (4)\\ &\quad + \mathbf{p}(\mathbf{z_c} \mid \mathbf{z_d}, \mathbf{z}^*, \neg\mathbf{h}) \, p(\neg h \mid z_d, z_d^*). \end{aligned}$$

The bold quantities here are defined in [4], sec. III B. The distribution over surface normal is similar to that over color. We calculate $p(h \mid z_d, z_d^*)$ as

$$p(h \mid z_d, z_d^*) = \frac{p(z_d \mid h, z_d^*)}{p(z_d \mid h, z_d^*) + p(z_d \mid \neg h, z_d^*)} = \frac{p(z_d \mid z_d^*)}{p(z_d \mid z_d^*) + p(z_d \mid \neg h, z_d^*)} \quad (5)$$

and analogously for $\neg h$. In [4] we modeled the uncertainty in the expected depth measurement, but not that in the observed depth. Including the observation uncertainty reduces the model's certainty (i.e. moves the output probability toward .5) in cases where the observed depth value is not very certain. This is useful both for points with large observed depth (which we mostly avoided using in [4]) and for points near depth boundaries, which can be misaligned in the image plane. In the model used in this paper we replace $p(z_d \mid z_d^*)$ in eqns. 3 and 5 with a marginalization over the observed depth z_d: instead of $p(z_d \mid z_d^*)$ we use $p(z_d \mid z_d^*, \sigma_d, \sigma_d^*)$, where

$$\begin{aligned} & p(z_d \mid z_d^*, \sigma_d, \sigma_d^*) \\ &= \int_{z_d^*} \int_{z_d} p(z_d \mid z_d^*, \sigma_d^*) dz_d dz_d^* \\ &= \int_{z_d^*} f_{exp}(z_d^*) \left[\int_{z_d} f_{obs}(z_d) p_{bin}(z_d, z_d^*) dz_d \right] dz_d^* \\ &\approx \int_{z_d^*} f_{exp}(z_d^*) \left[B f_{obs}(z_d^*) \right] dz_d^*, \end{aligned} \quad (6)$$

where we have made the dependence on uncertainty information explicit: σ_d and σ_d^* are the standard deviations of z_d and z_d^* respectively. $p(z_d \mid z_d^*, \sigma_d^*)$ is what [4] referred to as $p(z_d \mid z_d^*)$. $p_{bin}(z_d, z_d^*)$ is an indicator function for whether the two depth values are quantized to the same value by the sensor: for bin size $B = 1$ mm (a camera-specific value), $p_{bin}(z, z') = \begin{cases} 1, & |z - z'| < \frac{B}{2} \\ 0, & \text{else} \end{cases}$. f_{obs} quantifies observation noise: we use it to average over values that might have been observed. Similarly, f_{exp} is used to average over possible values of the expected depth. Both are modeled as Gaussians: for pixel p, $f_{obs}(z) = \mathcal{N}(z \mid z_d(p), \sigma_d(p))$ and $f_{exp}(z) = \mathcal{N}(z \mid z_d^*(p), \sigma_d^*(p))$. We compute $\sigma_d(p)$ using alg. 1, which takes nonlocal depth information into account to increase depth uncertainty near depth boundaries and near pixels with invalid depth readings. For now we compute $\sigma_d^*(p)$ using the same algorithm; ideally we would compute depth uncertainty from the volumetric map using a custom rendering algorithm.

Algorithm 1 computation of frame depth uncertainties using nonlocal information. σ_{max} is an uncertainty value assigned to invalid pixels; we use 1m.

```
for each pixel $p$ do
    $\sigma_d(p) \leftarrow StereoNoiseAtDepth(z(p))$
$I \leftarrow$ the set of pixels with invalid depth values or large depth discontinuities
Compute the Euclidean distance transform $D(p)$ of pixels to $I$
for each pixel $p$ such that $D(p) < 3$ do
    $\alpha \leftarrow \frac{D(p)}{3}$
    $\sigma_d(p) \leftarrow e^{\alpha \log(\sigma_d(p)) + (1-\alpha)\log(\sigma_{max})}$
```

IV. Online Change Detection

Since change detection can operate on a map and a single frame, we can run it online between a previously built map and each frame of a new scene. This can be used to aggregate change detection results on a map of the new scene that is built after the video is taken, as was done in [4], but it can also be used to separate our map of the scene into components for the static background and moved objects. Upon receiving each video frame, we update a map of the background and models of all movable objects, both those that appear in the new video but not previous videos and those that appear in previous videos but not the new one.

The background is represented as a volumetric map, as is each object. We use the Patch Volumes SLAM system [2] for aligning and adding RGB-D frames to volumetric maps. Patch Volumes represents a scene as one or more truncated-signed-distance-function (TSDF) volumes. Each such *patch volume* consists of four voxel grids that store the signed distance to the nearest surface, a measure of certainty of the TSDF value, the color in each voxel, and a measure of certainty of the color. The certainty weights allow for efficient depth map fusion using the technique of Curless and Levoy ([15]) on the GPU (as introduced by [1]). A new frame is aligned against this surface prediction by iteratively minimizing a dense error function incorporating depth and color information. The volumetric representation allows for parallel ray casting for surface prediction given a camera pose, and the current scene model can be extracted at any time as a colored mesh. Patch Volumes runs mostly on the GPU and can extend the size of the modeled volume as necessary. For efficiency in this paper we use statically sized volumes, although we do resize volumes periodically for object modeling.

Fig. 1 includes an example of change detection. Figs. 1(a) and 1(d) show frames from two scenes, and figs. 1(c) and 1(b) show change detection results of each scene with respect to the other.

V. Online Mapping and Segmentation

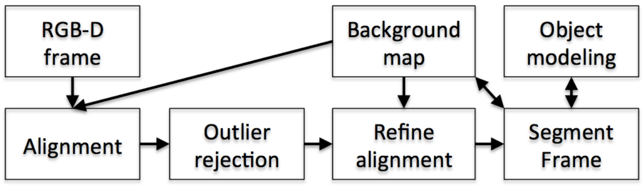

Fig. 2. our algorithm for processing each frame of a new scene.

The techniques of sec. IV allow us to create 3-D models of objects that move while the robot is away from the scene. At a high level, this is done by maintaining one map for each spatially connected region of surface that change detection marks as changed, and adding nearby pixels to these maps

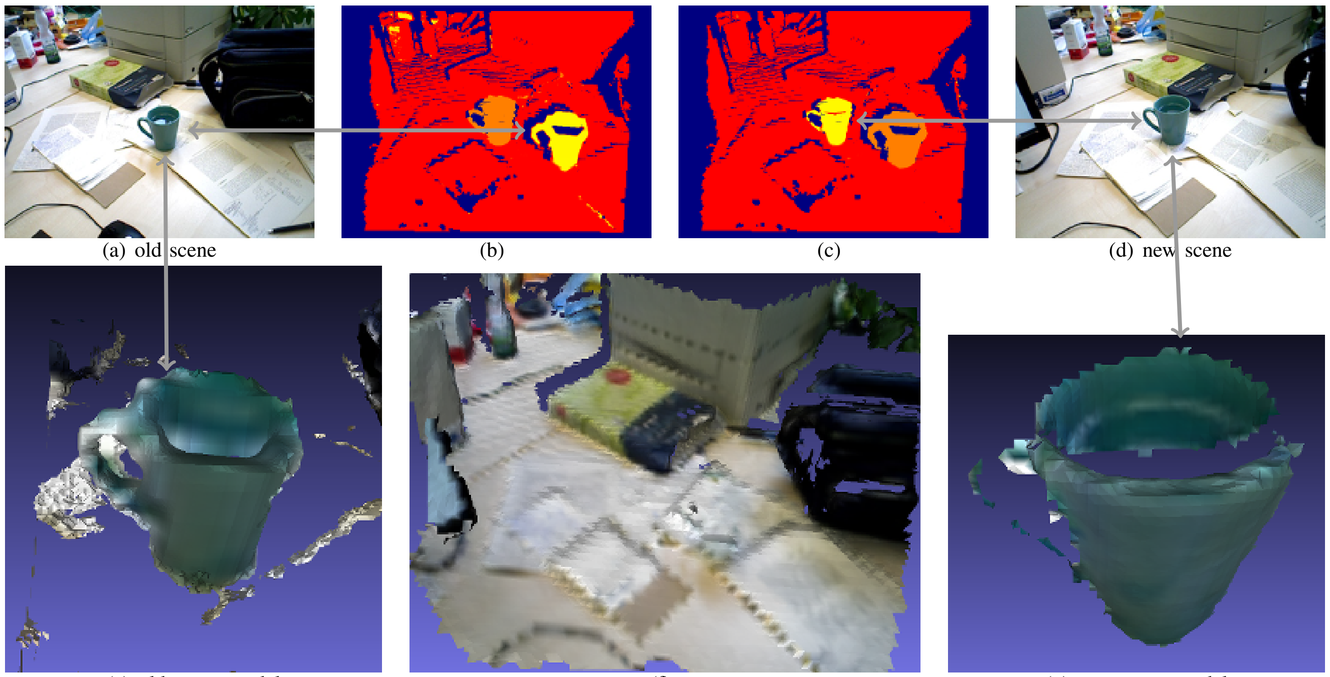

Fig. 1. (a), (d) two scenes of a desk on which a cup has moved; (f) the background map and (e), (g) object models resulting from running online change detection on them. (b), (c): change detection results for the first frame of the second video with respect to the map of the first scene and vice versa. Yellow regions are present in one scene but not the other; orange regions are occluded or uncertain; red regions are judged unchanged. Dark blue denotes no information at that pixel in at least one of the two change detection inputs.

as the video progresses according to a measure of how likely each pixel is to be part of the object modeled by each map. We create new object maps for pixels that don't match well to any existing object.

We can obtain the initial background for processing a video either from an empty map (all voxels are considered "unseen") or from the result of processing one or more previous scenes. We must initially align the video to the background map. This is done with point feature matching as in [4] but with the second map replaced by a single frame, and we ignore incoming frames until we have a good match. From then on, we process each frame with the procedure shown in fig. 2. We use the pose of the previous frame as an estimated pose for differencing the frame against the previous map (the result of running our procedure on zero or more previous visits to the scene) to perform outlier rejection by identifying pixels likely to be part of movable objects. We mask out those pixels, realign with Patch Volumes to get a more accurate alignment, and add the frame to the background map. Although Patch Volumes downweights likely outliers during alignment, we find that our outlier rejection works much better in practice. Except for runtime considerations we could run outlier rejection and realignment alternately to convergence. Next we run change detection again with the updated alignment to update the set of pixels that need to be explained with objects (i.e. outlier pixels with respect to the background scene motion). These pixels are segmented using information from both the current frame and object models we have already made, and each segment is added to one or more existing object models or used to create a new model. We run change detection in both directions to model both newly seen and no-longer-seen objects. After processing an entire scene in this fashion, we have a background map containing none of the objects seen to move as well as a model of each moved object seen in the current scene or in the background model used to process this scene. The assignment of pixels to models is one of the most computationally heavy parts of the system. Each existing model is rendered in the frame. Pixels to be added are segmented using 2.5-D connected components (alg. 2), and each segment is run through change detection with respect to each model. (This segmentation of the frame reduces the noise concomitant with choosing a model for each pixel separately.) A segment is added to each model that it matches well according to a change-detection-based score: for image segment s and existing model $\mathcal{M}$, we add s to $\mathcal{M}$ if $J_{s,\mathcal{M}} > \tau$, where

Algorithm 2 the 2.5-D connected components algorithm used to do per-frame segmentation prior to adding segments to models. δz_0 is the base distance threshold; we use 8 mm.

```
$\mathcal{P} \leftarrow$ the set of pixels with high probability of change
initialize a component for each pixel in $\mathcal{P}$
for each pixel $p \in \mathcal{P}$ do
    for each pixel $q \in \mathcal{P}$ within Manhattan distance $w$ of $p$ do
        $d \leftarrow$ the Manhattan distance from $p$ to $q$
        if $|q_z - p_z| < \delta z_0 \times d$ then
            connect the components containing $p$ and $q$
```

$$J_{s,\mathcal{M}} = \begin{cases} \frac{\sum_{p \in s} e_p (\frac{1}{2} - p_p(m))}{\sum_{p \in s} e_p}, & \sum_{p \in s} e_p > 0 \\ 0, & \sum_{p \in s} e_p = 0 \end{cases} .$$

Here e_p is Boolean and is 1 iff the sum of the magnitudes of all log-probabilities used in change detection is nonzero at pixel p (i.e. if there is any evidence at the pixel), and $p_p(m)$ is the probability of change at pixel p. The value of J is in $[-.5, .5]$; we apply a threshold $\tau \in [.3, .4]$ that depends on the parameters of the change detection model and the amount of noise in the maps (which is a function of map resolution and alignment error). If a segment is added to multiple models, those models are merged (and resized if necessary). The merge operation relies on the form of the volumetric maps: each voxel stores a signed distance and a weight that is essentially the number of times the voxel has been viewed. Merging two volumetric maps thus consists of adding the weights at each voxel (with one map represented by interpolated samples, since maps' axes aren't in general aligned) and performing a weighted sum of the two signed distances and colors at each voxel.

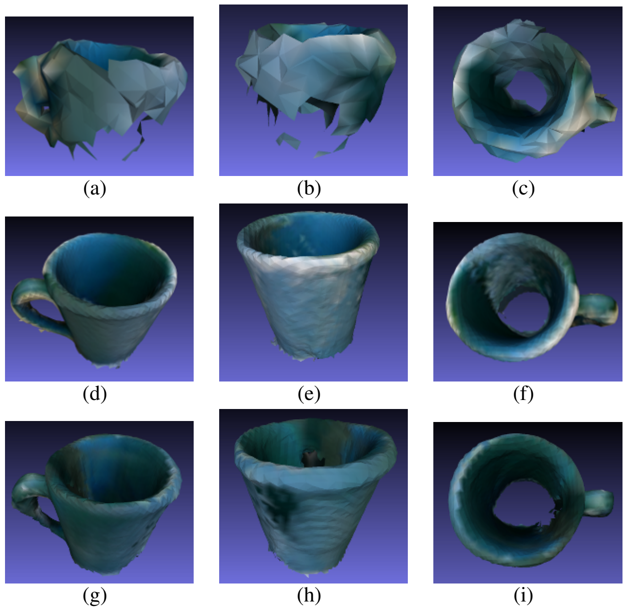

Fig. 3. three models of a blue cup made by processing the same scenes with different modeling options. Each column shows all models from approximately the same viewpoint. First row: 1-cm map resolution. Second row: 3-mm map resolution. Third row: 3-mm map resolution and improved per-pixel color confidence weighting when adding frames to the map.

Fig. 1 gives an example of the results of processing two scenes in this fashion. We start with an empty map, run the scene of fig. 1(a) through the above procedure to get a new "background" map containing exactly the contents of this scene, then run the scene of fig. 1(d) through this procedure to end up with the background map shown in fig. 1(f). The two scenes are represented here by the first frame of each video; each scene is hundreds of frames. The areas behind the object in each scene appear in the background map. While processing the second scene, we also find and model the object of fig. 1(e), which is in the first scene but not the second, and that of fig. 1(g), which is in the second scene but not the first. These are in fact the same object in different positions.

This system does not yet include *object discovery*, the clustering of views of objects in different scenes into groups corresponding to physical objects. However, we have worked on object discovery and intend to integrate that component with this system. Since our object discovery relies on high-accuracy models, we want to model all objects in as much detail as possible. For any algorithm that maintains a large number of maps, memory is a bottleneck. Since we have approximate segmentation information for objects, we can restrict the sizes of the maps we use to represent them. Because these maps are so much smaller than that of the full scene, we can afford to make them much higher-resolution than the scene map.

We show the result of processing many scenes in fig. 4. Each of six scenes was run through our system in order. The resulting background map is shown in the middle, along with some of the object models. We have used these results for a comparison with the batch object segmentation method of [14]. There are 36 unique object views in these six scenes. The batch method finds each of the 36 and creates an additional 45 non-object models, for a precision of 40%. The method of this paper creates models for 24 of the 25 views that it should find (an object that doesn't move until scene 4 of 6 can't also be segmented in scenes 1 and 2 with an online method); it also creates 12 duplicate models and 38 non-object models, giving a precision of 32% if duplicates are considered false positives or 49% if they're considered true positives. Images of the models are online; see sec. VIII.

We can control the visual quality of object models in a variety of ways. The first two rows of fig. 3 show the effect of changing the resolution of object models. The third row demonstrates a weighting scheme for adding information to the map in which the colors of pixels in the frame are downweighted according to their distance from depth discontinuities in the frame when being added to the map. This downweighting alleviates problems caused by depth and color image misalignment in the input RGB-D frames, which occurs even when the camera is stationary due to sensor noise and imperfect calibration. The tradeoff we observe is that the more accurate the model color, the less accurate the shape. The models in fig. 3 were generated from two videos, one of an empty surface and one of the surface with the cup on it, in which the camera viewed the object from almost all viewpoints, demonstrating that even using a SLAM algorithm without loop closure, we can make a complete and accurate object model. (The Patch Volumes framework includes loop closure, but it optimizes constraints between volumes rather than between frames, as our use case would require.)

As mentioned before, we have the option of using an empty map as the initial background map when processing a scene. One interesting effect of using this option is that we can process non-static scenes. If an object in the scene disappears *while the camera is pointing elsewhere*, then when the object's previous location is once again in the field of view we can detect it as moved and build a model of it; similarly if an object appears while the camera points elsewhere we can model it once we see it. An example

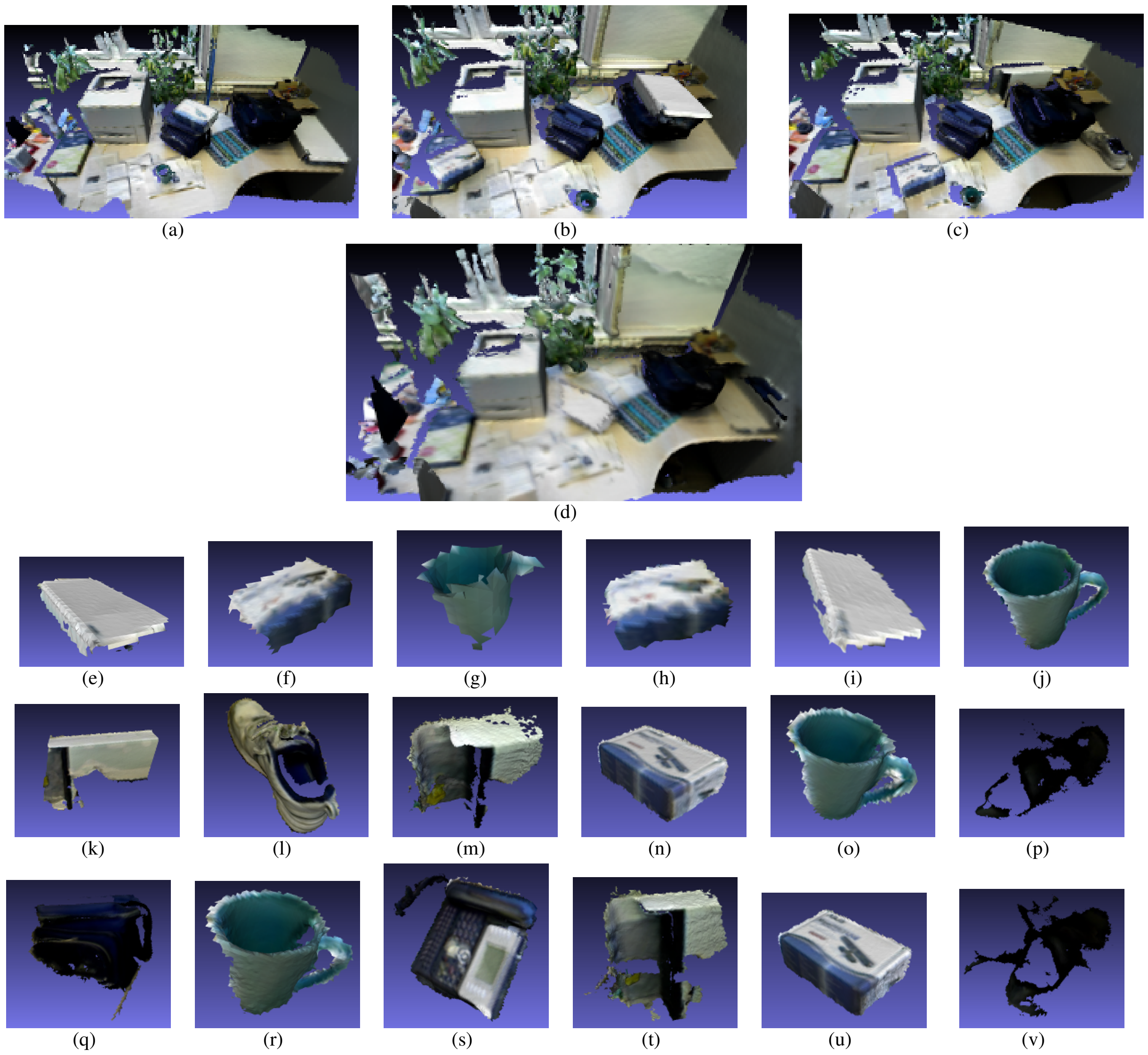

Fig. 4. (a) - (c) three of six scenes of an office environment encountered in sequence (the rest can be found on our website; see sec. VIII); (d) the background map obtained by processing all six scenes; (e) - (v) some of the object models created. For space reasons we show only the 18 models that represent actual objects; there are an additional 16 that duplicate objects shown here (in the same scene; we show each object once per scene we segment it from) and 38 models of non-object regions caused by noise. Here we see five views of a large textureless box, four of a textured box, four of a cup, one of a phone, one of a lunchbox, one of a clearly visible shoe and two of the shadowed shoe from those scenes where it is under the desk (lower right corner of scene).

is given in the accompanying video. In this mode, rather than differencing against a map created only from previous scenes, we difference against the current background map. Because it takes many frames for surface locations to change in volumetric maps, an object that appears or disappears must be viewed in many frames for enough evidence to accumulate in the map for previously present surfaces to be overwritten.

VI. Segmentation Disambiguation

Change detection has some undesirable characteristics for the purpose of object segmentation. Because there is no data association step in change detection, two objects that move between two scenes but that are physically close in one scene can't be segmented separately in that scene, even if they're separate in the other. Also, as mentioned before, if we see an area of space that we didn't see in a previous visit, we can't get change detection information for it. Both these issues affect the assignment of points to object models. If objects are close in 3-D and both are marked as changed, we will assign all points from both to a single object model. In the case of viewing a previously occluded area, we currently choose to add all previously occluded points to all nearby object models. In addition to these risks of obtaining an undersegmentation, we are also prone to oversegmentation. This happens when some points in between changed points have no change detection information due to an invalid depth in either the background or the current scene. If we could model all areas of the map in high resolution, we could assure undersegmentation by adding all points to a single model, to be split later once we have more information, but as it is, whatever heuristic we use to assign points to object models, we cannot guarantee that our models will represent either an

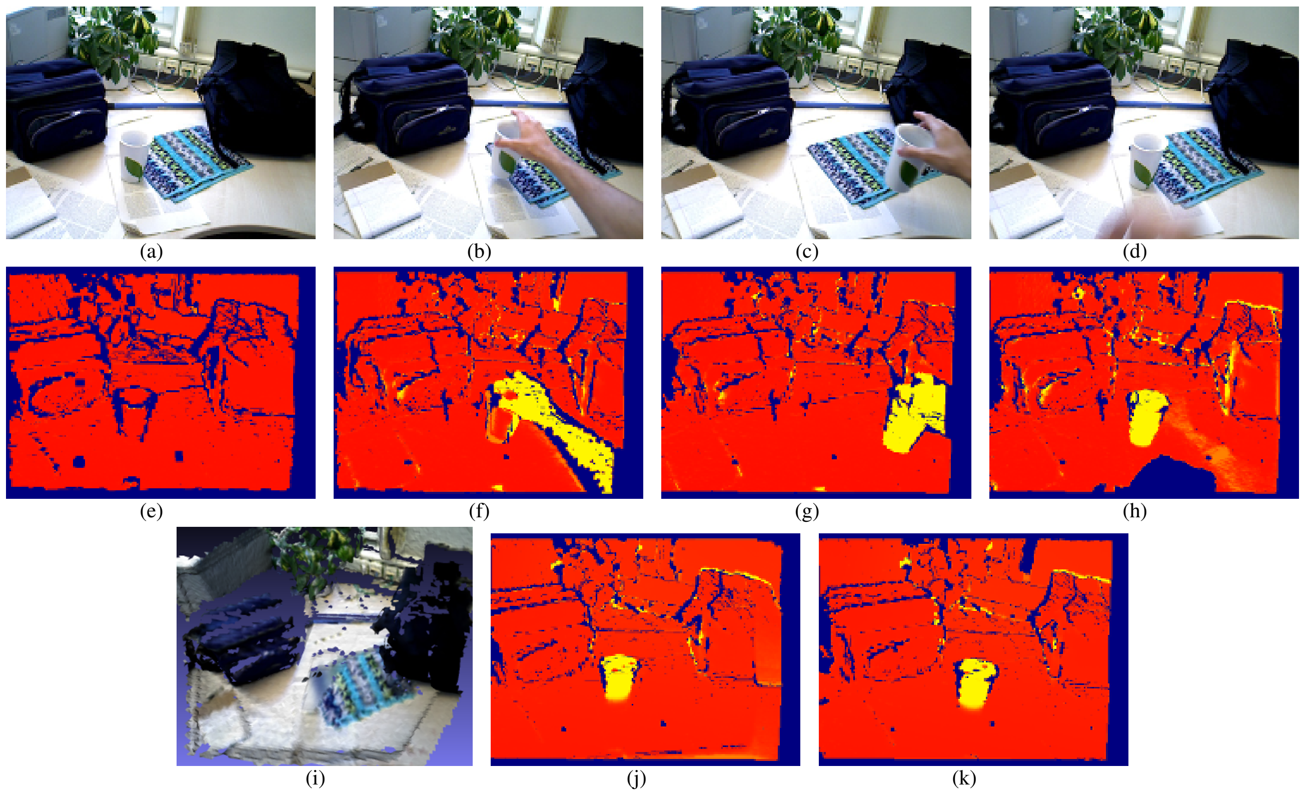
Fig. 5. (a) - (d) frames from the beginning, middle and end of a segmentation demonstration; (e) - (h) the result of differencing these frames against the demonstration background map at each time; (i) the background map at the end of the demonstration; (j), (k) the result of differencing the first and last frames of the demonstration against this background map. The object is nicely segmented.

over- or an undersegmentation of the actual set of objects.

As an example of how to acquire object segmentation information from sources other than change detection in static scenes, we here show how to use a human demonstration of object boundaries (hereafter “demonstration”) to provide the 3-D extent of a single object in the scene, as well as that of the empty space between the object and the background. The only user interaction required is picking up an object and putting it back down. In order to achieve this level of autonomy, we must place some restrictions on the interaction. The user should move one object out of the space it previously occupied, then replace it close to its original location (with a parameterized maximum distance). There can be camera motion during the demonstration. To ensure that demonstrations always provide unambiguous segmentations, we require that the object be put down in front of all possible occluders from the point of view of the camera during that demonstration (which does not preclude other objects from moving in front of that object’s location either before or afterward). If the system were running on a robot, the robot could alternatively attempt to grasp and move objects itself to learn their shape. However, this would entail potentially many failures to move the objects properly and would make the associated vision problems harder.

We determine the start of a demonstration to be a frame in which we see a large spatially connected segment of pixels that have moved with respect to a recent frame (e.g. three frames prior) and whose motion cannot be explained by the background motion computed by map alignment, and we designate an end to the demonstration when the number of changed pixels with respect to the pre-demonstration map ceases to change. At each demonstration frame, we identify changed points with respect to the current background map, and add all *other* points to the background map. This means that we add both pixels that agree very well with the map and those that were previously occluded or unviewed. We also find that we need to decrease the confidence in the TSDF each frame before adding to the map, or the map changes extremely slowly. (This is a particular problem during demonstrations.)

Once we have determined that the demonstration has ended, we segment out the object in the first and last frames of the demonstration. Assuming that at some point the object moves entirely out of the space it previously occupied and that no part of the user is visible at the start or end of the demonstration, this can be accomplished simply by detecting change in these two frames with respect to the background map, which no longer contains the object. We reoptimize the relative pose between the first frame and the background map at this point to correct for drift during mapping, which occurs both due to the lack of loop closure in our volumetric mapping and because we are abusing a static SLAM system to build a map of a changing environment, in that the modified frames we add to the map during a demonstration do not all contain the same surfaces.

Finally, using these segmentations we estimate the rigid motion of the object during the demonstration. We accomplish this with a RANSAC algorithm, repeatedly sampling triples of point correspondences between frames just before and just after demonstration, when we know the human is not present in the scene. We use scene flow to enumerate possible point correspondences, requiring only tens of RANSAC iterations. Due to the limitations of the image-based spatial regularization used in recent scene flow techniques, this algorithm requires a limit on how far the object can move during the demonstration.

At the end of the demonstration we create a volumetric map from the segment of the object in the final demonstration frame. (We assume the object being demonstrated was unknown to us before and so is not already modeled.) It would be possible to also add the object as seen in the first demonstration frame, since we have the rigid transformation between the two. It would not be possible to incorporate intermediate frames into the model due to the difficulty of segmenting the object apart from the human. If this demonstration component were integrated fully with our mapping system, object models resulting from demonstrations would be used in the same way as models acquired otherwise.

Fig. 5 visualizes a demonstration. The first two rows show some frames from a demonstration video and the change detection results for these frames with respect to the background map (fig. 5(i)), which changes significantly as the object being demonstrated moves. Initially (fig. 5(a)) the object, a mostly textureless cup, does not show up as changed. As it is moved, more and more of it is distinguishable from the background, and once it is set back down close to its original location, the background map no longer contains the object and it shows up as changed (fig. 5(d)). Figs. 5(j) and 5(k) show our final segmentations of the object in the first and last frames of the demonstration.

VII. Efficiency

Our system is intended for interactive use, so efficiency is a concern. Runtime depends on the number of object models that must be processed at each frame. Currently, when the number of models is under twenty, processing a frame takes .7 to 2 seconds. More than half of this is in change detection, the only major component of the system that does not run on the GPU. Change detection can be made to work on the GPU, and has not to date due mainly to time constraints. Our scene flow, which we use in analyzing segmentation demonstrations, runs on the GPU. It takes one to five seconds per frame pair, depending on scene content: the less visual texture the scene has, the more iterations of smoothing are required.

VIII. Conclusions

We have shown fast and accurate reconstruction of multiple rigid objects online during scene reconstruction. This allows us to represent a scene with one 3-D model of the static background as well as a model of each object we ever observe to have moved. We intend to integrate information about segmentation from the sources discussed here with other sources such as 1) detection of previously learned partial or full object models, using techniques similar to those of [16] but with our high-accuracy sensor model, and 2) active exploration by a robot. Additional images can be found at `http://www.cs.washington.edu/research-projects/robotics/rgbd-segmentation-online/`.

Robust online segmentation and object modeling is a prerequisite for tracking previously unknown objects, allowing object-level understanding of, for example, egocentric video [17] involving textureless objects. For tracking partially built object models through such video we might use dense alignment methods based on those we use in this work, change-detection-based methods similar to [16], or modifications of depth-only methods involving point matching between frames (e.g. [18]).

References

[1] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[2] P. Henry, D. Fox, A. Bhowmik, and R. Mongia, "Patch volumes: Segmentation-based consistent mapping with rgb-d cameras," in *International Conference on 3-D Vision (3DV)*, 2013.

[3] T. Whelan, M. Kaess, J. Leonard, and J. McDonald, "Deformation-based loop closure for large scale dense rgb-d slam," in *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[4] E. Herbst, P. Henry, X. Ren, and D. Fox, "Toward object discovery and modeling via 3-d scene comparison," in *IEEE International Conference on Robotics & Automation (ICRA)*, 2011.

[5] P. Fitzpatrick, "First Contact: an Active Vision Approach to Segmentation", in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[6] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun, "Towards object mapping in dynamic environments with mobile robots," in *International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[7] D. Haehnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *IEEE International Conference on Robotics & Automation (ICRA)*, 2003.

[8] R. Finman, T. Whelan, M. Kaess, and J. Leonard, "Toward life-long object segmentation from change detection in dense rgb-d maps," in *European Conference on Mobile Robots*, 2013.

[9] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, and A. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[10] P. Alimi, D. Meger, and J. Little, "Object persistence in 3-d for home robots," in *ICRA Workshop on Semantic Perception, Mapping and Exploration*, 2012.

[11] K. Wurm, D. Hennes, D. Holz, R. Rusu, C. Stachniss, K. Konolige, and W. Burgard, "Hierarchies of octrees for efficient 3-d mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[12] S. Pundlik and S. Birchfield, "Motion segmentation at any speed," in *British Machine Vision Conference (BMVC)*, 2006.

[13] A. Roussos, C. Russell, R. Garg, and L. Agapito, "Dense multibody motion estimation and reconstruction from a handheld camera," in *International Symposium on Mixed and Augmented Reality*, 2012.

[14] E. Herbst, X. Ren, and D. Fox, "Rgb-d object discovery via multi-scene analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[15] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *ACM SIGGRAPH*, 1996.

[16] M. Krainin, K. Konolige, and D. Fox, "Exploiting segmentation for robust 3-d object matching," in *IEEE International Conference on Robotics & Automation (ICRA)*, 2012.

[17] Second IEEE workshop on egocentric vision at CVPR 2012. http://egovision12.cc.gatech.edu/.

[18] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds," in *IEEE International Conference on Robotics & Automation (ICRA)*, 2013.