Object Segmentation from Motion with Dense Feature Matching

Evan Herbst  Xiaofeng Ren  Dieter Fox

Abstract—To enable fast deployment and long-term operation, service robots must have the capability to discover and learn about novel objects even in previously unseen parts of the environment. Since most mobile robots are interested in objects for their potential for manipulation, one good way for a robot to find objects is to attempt to manipulate the scene, then decide what parts of the scene the manipulation affected. We use motion as a cue to segment objects and register views of them before and after an attempted manipulation. Many common objects are not distinctly textured, so we match local descriptors for densely sampled points rather than sparse keypoints. Our method performs both dense segmentation and motion estimation for objects undergoing full 3-D rigid motion.

I. INTRODUCTION

An indoor service robot such as a robotic butler, upon being introduced into a new environment, should start keeping track of various geometric and semantic information about its surroundings. For example, it should always know the set of objects in its vicinity. It’s reasonable for the notion of an object to depend on the capabilities of the robot, so that a robot consisting of a camera and a manipulator arm, for example, is only interested in objects it can both see and manipulate. Our robot has a hand on the end of an arm, so for this paper we think about objects we can move with the hand. Once we are using movability as a measure of objectness, it is natural to use motion as a cue to find objects. Assuming the environment is also populated by humans, objects will move over time, so that the robot can leave a scene (local piece of the environment), return some time later, and find some objects by running a passive change detection algorithm. However, there will always be some objects that humans don’t move very often or that rarely appear without surrounding clutter. These scenarios motivate active object segmentation, in which the robot literally takes matters into its own hands and causes objects to move.

For purposes of this work we assume the robot is able to push surfaces in the environment until it detects that something has moved. Active exploration for finding objects is not a solved problem; recent approaches all start by fitting a table plane or assuming the background is a single color. However, we focus on segmentation and pose estimation of the moved object(s). We use RGB-D camera frames from before and after the movement, but not during; using all frames would allow us to employ tracking methods, but would require extra work to track the robot arm (since the arm’s reported pose matches its actual configuration only roughly [1]) or the arm of the human causing the motion.

G. Herbst and D. Fox are with the University of Washington Department of Computer Science & Engineering, Seattle, WA 98195. X. Ren is with ISTC Pervasive Computing, Intel Labs, Seattle, WA 98195.

Given two views of a scene containing objects that undergo arbitrary 3-D rigid motion, some sort of local matching is required to segment objects. If all objects are highly and distinctly textured, descriptors may only have to be computed at sparse keypoints. Our objects are not all highly textured, so we choose to match descriptors at densely sampled points. One natural set of dense points is the set of all samples we have of the scene, and this is the set we choose to use in this work. We propose a random-sampling algorithm to enumerate possible object motions and identify those most consistent with the scene. The algorithm works very well for well-separated objects despite the high percentage of false point correspondences considered. We can use the estimated motions either to immediately produce a segmentation or to derive a measure of our uncertainty about the correct segmentation, which we can use as a termination condition in an iterative segmentation process.

II. RELATED WORK

Segmentation of object instances, as opposed to classes, is usually performed on a large set of object views. For example, Kang et al. [2] acquire object instances from a large set of images by iteratively clustering segment hypotheses and refining the outlines of these hypotheses. However, a robot acquiring knowledge while performing its normal duties cannot afford to use batch methods. Therefore in this project we attempt to extract as much information as possible from a single episode of objects moving.

Change detection after such a single episode has been used for instance segmentation before. Bhat et al. [3] use point-feature matches to suggest 3-D rigid motions from 2-D images, then segment pixels in each frame separately using graph cuts. In active segmentation, Kenney et al. [4] push objects in a plane with a fixed overhead camera and track 2-D point features. They get sparse motion information from change detection between each consecutive pair of frames, and use this to accumulate evidence for motion over time. They use single-frame MRF inference after the push to provide a segmentation. Fitzpatrick [5] uses change detection on images taken before and after a push rather than feature matching, and also runs MRF inference to get a single-frame segmentation afterward. His method can find only a single moved object. Katz et al. [6] push objects in 3-D and track 2-D point features. They use the points’ 2-D motion to cluster features into multiple rigid bodies. They also use a single-frame segmentation afterward. Chang et al. [7] push a pile of multiple objects to segment it into individual objects. They also match point features before and after pushing, and use a learned mapping from the number
of geometrically consistent feature matches to singulation confidence to decide when confidence in object extent is high enough to stop pushing. All these active segmentation approaches assume an initial segmentation of the objects to be pushed—obtained in most cases by fitting or assuming a known location of a planar supporting surface—and all but [5] assume visual point feature matches are available. We avoid both of these strong assumptions.

In previous work [8] we introduced a change detector for RGB-D scenes given a global alignment of the scenes. It incorporates depth, color and surface orientation cues as well as modeling occlusion. As we will discuss later, from this model we can derive a geometric consistency score for a hypothesized pose of a set of surface samples with respect to another scene. We have shown [9] that this score is a better measure of geometric consistency than various scores derived from ICP [10]. Here we use this scene differencing operation both to identify surfaces that move during a push and to score candidate motions. This work improves on the segmentation method of [8] by using motion fitting to help locate object boundaries. [9] describes a method for matching objects at different times once the objects have been segmented; this work instead takes the approach of segmenting each scene and matching objects across scenes simultaneously, with the happy side effect of also estimating each object’s motion, which [9] was unable to do.

III. MOTION DETECTION FOR ALIGNMENT

In [8] we introduced a probabilistic measurement model for RGB-D cameras and used it to detect changes between two scenes \( S_0 \) and \( S_1 \). In this paper these scenes are snapshots taken before and after a hypothetical robot attempts to manipulate the environment (e.g. by pushing a selected surface point). \( S_i \) is represented as a set \( M_i \) of surface samples, which might be an RGB-D frame or an aggregated map, plus a set of camera views \( \{ V_{ij} \} \), each of which also consists of surface samples. We use a view-based formulation of change detection to account for occlusion. For each sample in \( M_i \) seen from each view \( V_{(i-j)} \) we estimate the probability that the surfaces represented by the sample and by the pixel it projects to in the view are the same. We use only very local information about the surfaces to make these judgments, in order to avoid smoothing information across object boundaries. Given a hypothesized alignment \( X \), we project each surface sample \( s \in M_0 \) into each view of \( S_1 \) to obtain the expected measurement \( z_s^* \), which consists of depth, color and surface orientation. Denote by \( z_s \) the surface sample from \( S_1 \) that projects to the same point on \( V_{1s} \)'s image plane as \( s \). A sensor model provides \( p(z_s | z_s^*) \).

Since we use the model for change detection, we actually have two sensor models, \( p(z_s | m_s, z_s^*) \) and \( p(z_s | -m_s, z_s^*) \), for the cases in which \( s \) has and has not changed. We denote the boolean variable specifying whether a surface sample has changed, or “moved”, by \( m_s \).

For this paper, each scene consists of a single RGB-D frame. We estimate the probability of change for \( s \) as

\[
p(m_s | z_s, z_s^*) = \begin{cases} 
  p(m_s | z_s^*)p(z_s | m_s, z_s^*) & \text{if } p(m_s | z_s^*) + p(-m_s | z_s^*) \neq 0 \\
  0 & \text{otherwise}
\end{cases} \tag{1}
\]

\[
p(m_s | z_s, z_s^*) \approx \begin{cases} 
  p(m_s)p(z_s | m_s, z_s^*) & \text{if } p(m_s | z_s^*) + p(-m_s | z_s^*) \neq 0 \\
  0 & \text{otherwise}
\end{cases} \tag{2}
\]

\[
p(m_s | z_s, z_s^*) \propto p(m_s)p(z_s | m_s, z_s^*), \tag{3}
\]

where the approximation in (2) is that the probability of an object moving doesn’t depend on its distance from the camera, and that in (3) is allowed because we only maximize over \( m_s \). The forms of the distributions we use can be found in [8].

From this change detection model we can derive a score for a hypothesized rigid alignment. Given a set \( U \) of surface samples in the first scene and a proposed alignment, we can run differencing at each point sample and aggregate some statistic over all surface samples to get a score. Here we choose to score a hypothesis with

\[
S(X) = \sum_{s \in U} \max(0, 0.5 - p(m_s | z_s, z_s^*)). \tag{4}
\]

The higher this score, the better the alignment. This score rewards surfaces that match well, avoids penalizing occluded surfaces, and avoids over-penalizing surfaces that match extremely badly, which is helpful when there are likely to be extraneous points among the surface samples.

In addition to computing this geometric consistency score for a given hypothesized pose, we can optimize the score. This objective is highly nonconvex, so initialization is necessary; for this work we will initialize the optimization using the highest-scoring of a large set of poses proposed by random sampling. This optimization is similar to that of ICP, but handles color in a non-heuristic way (as opposed to, e.g., the popular methods of running ICP in a combined space of position and color and of restricting possible correspondences using color distance thresholds), does not require choosing point correspondences, and models occlusion properly. We set up the optimization as nonlinear least squares and optimize with Levenberg-Marquardt (LM). LM has a minimum required number of “measurements” (of which the objective is the sum of squares), but the more measurements are used, the slower the optimization. We optimize 6 parameters—translation, roll, pitch and yaw—so we divide the surface samples into N groups; each of N “measurements” is eq. 4 summed over the corresponding group of samples. Using each surface sample as a separate measurement might yield a better optimization objective, but with \( N = 16 \) the optimization already takes multiple seconds for even moderately sized clouds (10,000 points).

IV. MOTION ESTIMATION

Pseudocode for our algorithm to identify the rigid motions that occur in the scene is shown in alg. 1. First (line 1) we run passive change detection over the before- and after-push frames; then (line 2) we identify the points that most clearly do not agree with the identity motion. These are the points whose motion we know we need to explain by proposing
Fig. 1: the result of scene differencing on two before-and-after-push frame pairs. This and future figures show scenes of household objects on a table, with the floor and chairs in the background. (e), (f) differencing results for the frames of (a) and (b), in which the object moves far enough that there is no overlap between its before- and after-push 3-D positions. (g), (h) differencing results for the frames of (c) and (d), in which there is overlap. The brighter yellow the differencing result at a point, the higher the probability that that surface changed. Orange points have high uncertainty of \( m_s \), usually because they are occluded in the other scene. Yellow points will become seeds for RANSAC.

**Algorithm 1**: our iterative rigid motion estimation algorithm.

```
input : RGB-D frames \( F_0, F_1 \)
1 Difference \( F_0, F_1 \) using the identity transformation
2 Select seed point sets \( S_0 \subset F_0, S_1 \subset F_1 \)
3 Compute descriptors and cross-frame descriptor similarities for seed points
4 while there are seed points unexplained by a transformation do
    Randomly generate many transformations \( X \) and score using eq. 4 with an approximation to \([8]\)
    Rank all transformations by approximate score
    Keep only the top \( k \) transformations and rerank them using eq. 4 with \([8]\)
7 foreach \( X \) by score do
    Find inlier points \( I_0 \subset F_0, I_1 \subset F_1 \) for \( X \)
    Iteratively reoptimize \( X \) to fit \( I_0 \) and \( I_1 \) and reselect inliers
    if enough inliers then
        Declare a new object and remove \( I_0, I_1 \)
        from the set of points to consider
    end
end
```

other motions. (There may also be points that agree well with the identity transformation but that we will want to explain otherwise.) For this reason and for efficiency, we use only these points to generate motion hypotheses to be evaluated. For this paper we define “do not agree” to mean

\[
p(m|z_s, z^*_s) \in [.05,.5−\epsilon] \cup [.5+\epsilon, 1],
\]

excluding points from one scene whose hypothesized position in the other scene is occluded (in which case \( p(m|z_s, z^*_s) = .5 \)), since we have no positive evidence that those points are not present in the other scene. We call such “non-agreeing” points seed points.

We use local RGB-D descriptors to select point correspondences from which to generate rigid transforms. Over each point’s 3-D neighborhood we compute a histogram of curvatures, a histogram of colors and a histogram of color gradients. If we expected relatively high texture in color or shape we might also use more distinctive descriptors such as spin images \([11]\) or a point-cloud version of orientation-invariant SIFT. Parallelized over 16 hardware threads, this takes about 5 seconds per frame on a 2.7GHz Intel Xeon X5550. We concatenate the histograms and compute the \( L_1 \) distance from each before-push seed point’s descriptor to each after-push seed point’s descriptor. On an NVIDIA GTX580 GPU, distance computation takes no more than 10 seconds when there are 30,000 seed points per frame. For memory reasons we subsample seed points in one of the two frames to 30%.

There are a variety of existing approaches to matching multiple objects undergoing rigid motion between two scenes. Gordon and Lowe \([12]\) compute SIFT descriptors for sparse keypoints and match them with the standard ratio test, which works when there are a large number of highly distinctive neighborhoods on all objects. If there aren’t individual distinctive neighborhoods but there are clusters of descriptors for sparse keypoints that are well separated at the cluster level, the method of Collet et al. \([13]\), which quantizes SIFT descriptors and requires only that each match be to a descriptor in the same bin, is applicable. However, we want
to be able to match objects like single-color dishware, for which no set of sparse keypoints will produce distinctive descriptors, assuming it’s even possible to choose sparse keypoints intelligently. Therefore our method generates and scores a large number of matches between densely sampled points that have high descriptor similarity.

To generate each motion proposal (pseudocode line 5) we choose one seed point from each frame and a rotation angle about the first seed’s normal axis. Together with a deterministically defined rotation about the second-frame seed’s normal, these uniquely identify a 3-D transformation. We score each transformation using an approximation to the score discussed in sec. III that uses only the depth component of the sensor model and is readily implemented on a GPU. We can run this approximate scoring on about 5000 hypotheses per second, as opposed to 75 per second with full scene differencing. In order to avoid running the full scoring operation on too many proposals, we rank proposals by approximate score and run the full model on only the top \( k = 100 \) at a time. This mini-batch structuring is a compromise between generating and processing hypotheses one at a time, such that we can stop as soon as most seed points have been explained by motions, and generating and scoring all possible hypotheses in one batch, such that we know the best-fitting hypotheses possible are in our list of options.

Once we have selected a short list of hypotheses to consider further, we run scene differencing over the seed points in each frame with respect to the other scene, and compute inliers as those points with low \( p(m|z_z) \). We then restrict the set of inliers to the connected component including the seed point. This heuristic is motivated by cases in which the seeds are taken from one object but the inliers are mostly on another object, as in fig. 2. It’s not clear whether to approach such cases by trying to figure out that the object containing the seeds doesn’t match well or by shifting our focus to the object with most of the inliers, so we instead cull proposals whose inliers aren’t near the seeds.

Each hypothesis in turn is refit to the inlier set using the optimization discussed in sec. III, after which (line 12) if there are enough inliers remaining we declare the inliers to constitute a new object. We then remove these inliers from consideration for future hypotheses, to counter the well-known proclivity of RANSAC methods to find only the largest consensus group. At the end of this iterative process, we have a set of motions each of which had a large number of inliers. In general there will be more candidates than true motions. For example, the inlier sets found for the frame pair of fig. 2 are shown in fig. 4; there are four objects and random sampling finds five motions. We therefore jointly select one motion at each surface sample in a further structured inference step.

V. SEGMENTATION

The inlier selection step of our random sampling algorithm gives us a dense assignment of motions to surface samples, so unlike [14], [5], [6], [7], we could use the result of this step as a final segmentation. However, our inlier selection is very heuristic, so we postprocess the short list of candidate motions to choose one at each surface sample. Selecting the best-fitting candidate at each point is an improvement over using the inlier sets to provide a segmentation, but we can do better with joint inference. For each surface sample in each frame (before and after) we create a node in a conditional random field, with a discrete variable specifying which candidate motion is selected at that point. We include two types of binary clique: spatial smoothness between neighbors in the same frame and temporal consistency between points that map to each other under one of the proposed motions. We use Potts potentials for the smoothness cliques; for consistency cliques generated from the \( k \)th motion proposal, we penalize only if one point selects motion \( k \) and the other does not. Such temporal consistency cliques have been used, e.g., for labeling points in laser scans of urban environments [15]; in that work only one motion was used to generate such cliques.

In sec. VI we show examples of using MAP inference in such CRFs to provide an intermediate segmentation of a scene after a single push, but more importantly also of using marginal inference to define a measure of local object-boundariness at each surface point. We use the boundariness measure to define a measure of segmentation uncertainty over the scene, and plan to use segmentation uncertainty to decide when to terminate an iterative active segmentation approach to understand all objects in a scene.
VI. Experiments

First we show an example of segmentation of a simple scene with a single moving nontextured object, in fig. 3. The motion is small enough that the initial and final object positions overlap, so differencing by itself doesn’t identify the whole object. Our full method, however, is able to identify the entire object both before and after the motion.

![Fig. 3: Segmentation results for the bowlMediumMotion dataset.](image)

We have also run the method on datasets containing multiple moving objects, such as that shown in fig. 4, which contains four objects close enough together to present problems to state-of-the-art static RGB-D segmentation algorithms. We find five motions, although the inlier sets for the motions don’t correspond to the same objects in the before and after frames. In particular, in the before frame the white bottle is included with the inlier set of the white bowl. To avoid this we probably want to replace the connected-components step in our inlier finding procedure with something with fewer hard thresholds. For this dataset we also show motion-derived boundariness. Unsurprisingly, the boundariness measure varies much less with CRF parameters than the MAP segmentation does. Since the uncertainty of the segmentation of the scene is what we really want for use in future processing, this robustness is good.

![Fig. 4: Segmentation results for the dataset shown in fig. 2.](image)

Finally, fig. 5 shows a segmentation and boundariness for a scene with multiple objects only one of which moves. This motion would be an ideal result in our future robot experiments: the robot manages to successfully push only
one object, and pushes it far enough that its initial and final positions don’t overlap, so that ideally the object can simply be picked out of the differencing results. In reality, we identify multiple motions and the regularization step fails to completely remove them.

![Fig. 5: Results for a dataset with multiple objects only one of which moves. (a), (b) the before- and after-push frames. (c) inliers in frame (a) for the different motions we identify (identity transformation in green). Segmentations from CRF MAP inference with different potentials: (d) unary potentials only; (e) unary, smoothness and consistency potentials. (f) the probability-of-boundary map (shown for one combination of clique types; all others are very similar).](image)

In general, temporal consistency constraints seem to be more useful than spatial smoothness constraints; in particular, we could play with different types of spatial aggregation on top of the temporal-consistency-CRF results and perhaps reduce oversmoothing. However, consistency constraints do cause a repetition effect, visible for example in figs. 3(g) and 4(g), caused by the interaction of points that have CRF cliques for multiple motions with the structure of our temporal consistency potential. The best way to avoid this problem would involve balancing the large number of cliques on some variables with other cliques for all other variables. Alternatively we might use only the segmentation uncertainty in future processing rather than MAP segmentations, and mostly avoid the problem.

VII. CONCLUSIONS

We demonstrate segmentation and registration of multiple objects undergoing 3-D rigid motion, without relying on table plane fitting or distinctive keypoints. The method makes heavy use of a change detection model that operates on surface samples. It typically runs in 1 - 4 minutes on a 16-core 2.7GHz CPU when each scene is one Kinect frame, and is not fully optimized, leading us to believe that matching of dense point sets can be accomplished reasonably efficiently, one of our goals.

Currently objects usually have to be well separated in 3-D for us to find them. This is mainly because of our inlier selection method; the use of connected components is reasonable when objects are well separated, but it’s less clear whether it will work for objects touching each other. Clearly the current method could not handle the scenario of a kitchen crowded by stacks of piled dishes. One insight into dealing with crowded scenes comes from Hager and Wegbreit [16], who point out that objects can be understood in “front-to-back” order of occlusion. In our context this means that we can iteratively focus on explaining some set of points that are not occluded by any others. At any point in time, most of the points in the scene are not very important to us; the challenge in using this insight is in figuring out which points we can afford to make mistakes on at what point in scene processing.

REFERENCES