2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Tracking-based Interactive Segmentation of Textureless Objects

Karol Hausman, Ferenc Balint-Benczedi, Dejan Pangercic, Zoltan-Csaba Marton, Ryohei Ueda, Kei Okada, Michael Beetz
Intelligent Autonomous Systems Group, TU Munich and Jouhou System Kougaku Laboratory, University of Tokyo
Email: {hausman, ferenc.balint-benczedi, pangercic, marton, beetz}@cs.tum.edu, {ueda, k-okada}@jsk.t.u-tokyo.ac.jp

***Abstract*—This paper describes a textureless object segmentation approach for autonomous service robots acting in human living environments. The proposed system allows a robot to effectively segment textureless objects in cluttered scenes by leveraging its manipulation capabilities. In our pipeline, the cluttered scenes are first statically segmented using state-of-the-art classification algorithm and then the interactive segmentation is deployed in order to resolve this possibly ambiguous static segmentation. In the second step the RGBD (RGB + Depth) sparse features, estimated on the RGBD point cloud from the Kinect sensor, are extracted and tracked while motion is induced into a scene. Using the resulting feature poses, the features are then assigned to their corresponding objects by means of a graph-based clustering algorithm. In the final step, we reconstruct the dense models of the objects from the previously clustered sparse RGBD features. We evaluated the approach on a set of scenes which consist of various textureless flat (e.g. box-like) and round (e.g. cylinder-like) objects and the combinations thereof.**

I. Introduction

A service robot operating in human environments may be required to perform complex dexterous manipulation tasks in a variety of conditions. For example, when setting a table [1] the robot is likely to be confronted with a cluttered unstructured scene[1] like the example shown in Fig. 1. In order to successfully perform this task, the robot must be able to detect the individual objects. Without the ability to interact with the environment, it is difficult to distinguish between the object boundaries and texture patterns, particularly in the presence of objects of similar colors, shapes and sizes.

To demonstrate this we tested three state-of-the-art segmentation algorithms operating in depth, RGB and RGBD space respectively on the given scene. The results are shown in Fig. 1. We notice that they are far from being optimal in the cases of a) same color objects (a coffee mug and a saucer), b) similar shape objects and occlusions (a white and a blue box), c) stacked objects (an egg and a plate) and also in the case of d) a sensor default (cutlery in this case appears transparent to the Kinect sensor). Following structure from motion approaches, one could observe the scene from various views and apply merging of hypotheses. This approach would however fail in the case of non-navigable spaces for the robot. While one can certainly fine tune the algorithms' parameters for a certain setup and environment, it is easier and arguably more natural to exploit the robot's embodiment and interaction capabilities in order to obtain a better understanding of its environment. Reaching out to get a sense of what is around is the way how infants get to know their "near space" according to Piaget's theory of spatial cognition in the sensorimotor stage (until the age of 2), and getting a hold of connectivity (i.e. object unity) is an important factor in the infant's understanding of objects at that stage [2].

[1]Following the discussion at the Clutter12 workshop at RSS 2012 we acknowledge that this is a "laboratory clutter" where the degree of difficulty is similar to the scenes from the related works but still inferior to the real world clutter.

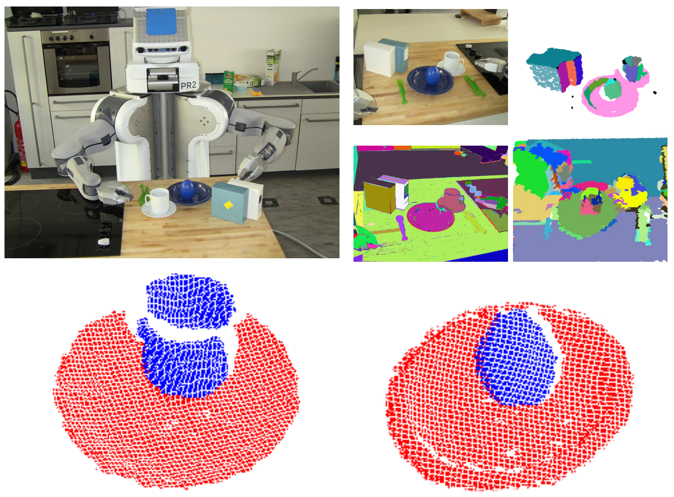

Fig. 1. Top-left: The service robot PR2 aiming to segment the scene consisting of textureless object. Results of the scene (top-right, NW) segmentation using Part-Graph-based Hashing [3] method (top-right, NE), Graph-based segmentation method [4] (top-right, SW) and the Region Growing method [5] (top-right, SE). These methods work in depth, RGB and RGBD space respectively and all underachieve due to the complexity of this challenging task. Bottom row: white mug on the white saucer (left) and blue egg on the blue plate (right) from this scene correctly segmented using the interactive approach presented in this paper.

Our approach: In this work we focus on proposing a solution for the cases a), b) and c) from above. Similar to Katz et al. [6], Bergstrom et al. [7], and our earlier work [8] we propose a system that uses induced motions in a scene to enable effective object segmentation. Our system employs a combination of the following complementary techniques: pre-segmentation of a raw point cloud of a given scene from a single camera view using part-graph-based hashing [3], estimation of a contact point and a push direction of the robot's end effector [8], RGBD feature extraction and tracking using particle filtering-based tracking, graph-based feature trajectory clustering algorithm, and dense model reconstruction based on region growing in normal space. There are three important assumptions in our system. First, that each item is a rigid body and not subject to large deformations when interacting with the robot's end effector or other objects. We also assume that the objects are either flat (box-like) or round (cylinder-like), which holds for most household objects in publicly available databases [9], and that in the tracking step the features do not get more than 50% obstructed.

The evaluation was performed on 17 scenes with challenging arrangement of flat and round objects of similar colors, shapes and sizes. 82% of objects were segmented correctly in these scenes. Our system is available as open source[2] and can be deployed on a robot equipped with either a 2D-camera and a depth camera or Kinect camera and at least one arm.

Overall, we present the following main contributions for the segmentation of scenes consisting of textureless tabletop objects:

- A set of RGBD features suitable for the tracking of flat and round textureless object (Sec. V-A);
- A graph-based algorithm for the clustering of 3D-feature trajectories, in which graph edges measure the dissimilarities between the RGBD features' distances (Sec. V-C);
- The inclusion of a static scene pre-segmentation algorithm and a probabilistic method for the detection of over or under-segmentation (Sec. III-B);
- A dense model reconstruction algorithm that makes use of the already clustered features (Sec. V-D);
- And the integration of all the above into a pipeline using the Robot Operating System (ROS[3]) as depicted in Fig. 2.

II. Related Work

Research in passive perception has traditionally focused on static images and segmented images based on a set of features such as color [10] or higher order features such as the ones found in graph cut approaches [11].

This paper focuses on interactive scene segmentation by adding robotic arm manipulation into the perception loop. Segmentation of rigid objects from a video stream of objects being moved by the robot has been addressed by Fitzpatrick [12] and Kenney et al. [13]. These works are based on the segmentation of objects from a video stream of a pre-planned arm motion, use a simple Gaussian model of the color values to infer the possible motion and a graph cut algorithm for the final object segmentation. These approaches can deal with textured as well as textureless objects. In contrast, our arm motion is not pre-planned but adapts to the scene and we make use of 3D data to segment the object candidates from the background.

Both approaches presented in this paragraph work with the textured objects only. Katz et al. [6] address the problem of segmenting the articulated objects. A Lucas-Kanade tracker and a set of predictors (relative motion, short distance, long distance, color, triangulation and fundamental matrix) are applied to obtain rigid body hypotheses (in form of a graph) and a subsequent fixation point on the object. The latter is used to segment an object based on color, intensity and texture cues. The major limitation of this approach is the pre-planned arm motion and the time needed to break the graph of object hypotheses into the subgraphs using a min-cut algorithm. Bergstrom et al. [7] propose an approach to interactive segmentation that requires initial labeling using a 3D segmentation through fixation which results in a rough initial segmentation. The robot interacts with the scene to disambiguate the hypotheses. Points in the motion space are clustered using a two component Gaussian mixture model. A limitation of the system is in that the number of objects per scene never exceeds 2.

Some approaches examine how the perturbations can be planned to accumulate a sequence of motion cues. Gupta et al. [14] use a set of motion primitives consisting of pick and place, spread, and tumble actions to sort cluttered piles of single-color objects. Euclidean clustering is used in the distance and the color space to classify the scenes as uncluttered, cluttered, or piled. Distance-based clustering is limited as its success is subject to correctly selected threshold. Color-based clustering may fail in the presence of sudden lighting changes. Additionally the system assumes that the objects (duplo bricks) are of a similar size. Chang et al. [15] present a framework for interactive segmentation of individual objects with an interaction strategy which allows for an iterative object selection, manipulation primitive selection and evaluation, and scene state update. The manipulation primitive selection step uses a set of heuristics to maximize the push action, however, it is unclear in how much this component contributes to the successful segmentation of the objects. The manipulation primitive evaluation step uses sparse correspondences from the Lucas-Kanade optical flow tracker and computes a set of transforms which are color matched against a dense point cloud. A likelihood ratio of a target being a single item or multiple items is determined based on the magnitude of the transform motion and the percentage of dense point matches. The major limitation compared to our work is that they do not estimate corner contact points.

There is a corpus of works dealing with the estimation of the articulation models for drawers, boxes, etc. [16], [17]. The common problem for both approaches is in that they assume the presence of a large, moving plane which they can reliably detect by running e.g. a RANSAC algorithm on the input point cloud and which unanimously represents the part of the object they are looking for.

III. System and Pre-processing

A. System Pipeline

Our approach consists of five main steps as depicted in Fig. 2 and demonstrated in an accompanying video[4]. In the first step we obtain an RGBD point cloud from the Kinect sensor. In the second step we perform static object pre-segmentation which results in a set of categorized object hypotheses O, with the category being either flat or round, and a list of object parts P_o that every object $o \in O$ consists of. Having obtained the object hypotheses O we infer which hypothesis is segmented correctly. For that we count the number of parts that the respective object hypotheses O consists of and then sample from the Poisson distribution

[2]http://www.ros.org/wiki/interactive_segmentation_textureless

[3]www.ros.org

[4]http://youtu.be/Bu4LayrGC1s

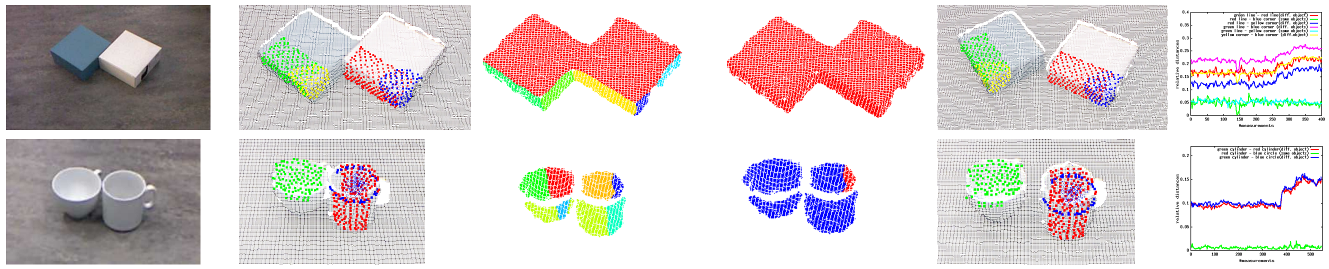

Fig. 3. Two test scenes in the top and bottom row respectively. First column: original scenes; second column: extracted RGBD features before the interaction; third column: parts P from the static segmentation; fourth column: object hypotheses O from the static segmentation; fifth column: tracked RGBD features after interaction; sixth column: relative distances between the tracked features. Plots with the ramp denote distances between features on different objects and plots with the constant values denote distance between features on the same object.

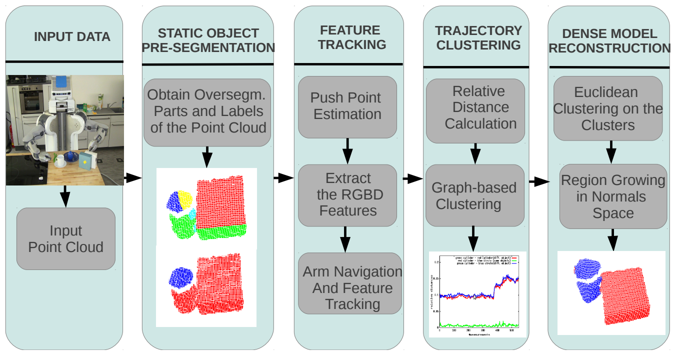
INPUT DATA
Input Point Cloud
STATIC OBJECT PRE-SEGMENTATION
Obtain Oversegm. Parts and Labels of the Point Cloud
FEATURE TRACKING
Push Point Estimation
Extract the RGBD Features
Arm Navigation And Feature Tracking
TRAJECTORY CLUSTERING
Relative Distance Calculation
Graph-based Clustering
DENSE MODEL RECONSTRUCTION
Euclidean Clustering on the Clusters
Region Growing in Normals Space

Fig. 2. System pipeline.

according to the Eq. 1. After obtaining the probability of the scene being segmented correctly we decide if the interactive segmentation algorithm should be used or not.

We use categorization of the objects as a prior for tracking by extracting and tracking line and corner RGBD features on the flat object hypotheses and circle and cylinder RGBD features on the round ones in the third step. Finally, we execute the arm motion movement in $1cm$ intervals until we reached a maximum of 5 pushes. All of the features are being tracked during the interaction and the trajectories of feature centroids are being saved. Based on relative distances between the feature centroids, the graph-based algorithm for the trajectory clustering is applied. The output of the algorithm is the number of objects belonging to a certain object hypothesis o and the association between the object number and the parts $p_1, \ldots, p_n \in P_o$ that belong to it (fourth step). In the fifth and the last step the dense model is reconstructed using the region growing algorithm where the tracked and clustered RGBD features are used as seed points.

B. Static Pre-segmentation of Objects

In order to achieve a pre-segmentation we make use of the classification method presented in [3] based on part-graph-based hashing. The basic idea is that segmenting objects accurately in a cluttered scene does not always yield the expected result, as seen in Fig. 3 column 4, and can lead to classification failures, but over-segmenting is easily realizable [18]–[20]. We use the classification approach described in [3] for categorizing over-segmented object parts in cluttered scenes by considering combinations of these parts to compute features and classify these efficiently with the aid of hashing. The result is a set of labeled parts with geometric categories that can be grouped in order to obtain object hypotheses. Based on statistics computed from the training data on single objects, we can estimate how likely it is that an object hypothesis is correct.

In the rest of the section we summarize the part-graph-based hashing algorithm briefly and show how we use it to guide the interactive segmentation.

1) Decomposition into Part Graphs: In order to find the parts $(p_1, \ldots, p_n \in P_o)$ in the point clouds we use the clustering criteria presented in [20], such that patches with a small curvature are considered, as shown in Fig. 3 column 3. For each part we subsequently compute GRSD- (Global Radius-based Surface Descriptor [21]) feature and store it for later use. We then extract the part neighborhoods by checking if the physical distance between two parts falls below a threshold of $2cm$ (considering Kinect noise level [22]), and build a connectivity matrix. Starting at each vertex of the connectivity matrix, we create all the possible groupings up to a certain size (eight parts in the case of single objects and four in the case of cluttered scenes) in order to obtain the "soup of segments", and create the groups' hash codes using isomorphic graph metrics. The hash codes are then used to further split the feature space ending up with a separate classifier (nearest neighbors in our case) for each hash code. During the classification phase we obtain confidence votes only from those classifiers, which were created for the hash codes that are found in our scene. Based on these votes a decision is made upon the class of the segments. For a detailed description of this approach please refer to [3].

2) Object Part Categorization: The classifier was trained on a subset of the dataset from [23] as presented in [3]. The choice of the feature determined for each part, namely the GRSD- is motivated by the fact that we are dealing with novel objects not seen before by the classifier, so in order to successfully categorize them we need to use geometric features. Additionally, the low dimensionality and additive property[5] make GRSD- a suitable choice for such task.

[5]If the feature is additive, the descriptor that would be computed for the object is the same as the sum of the features of its segments.

Objects ($o_1, \ldots, o_n \in O$) are categorized in six geometrical categories: sphere, box, rectangular/flat, cylinder, disk/plate and other. Doing this we get a better a discrimination between different objects. After having the results for the six geometrical classes, we merge them together into different *object types* considering everything spherical and cylindrical being *round*, and disks/plates, flats and boxes as *flat* objects. With the category *other* we thus get three object types, whereas most household objects fall into the first two [9].

In this paper we omit the category *other* and use the other two in order to determine if the interactive segmentation is needed, and if yes, which RGBD features to extract and track in the respective part of the point cloud in the given scene.

C. Verification of Correctness of Segmentation

Since the geometric categorization of parts does not give the correct grouping of these parts to form objects, simply grouping the parts of the same category together does not always separate the objects, especially if classification errors occur too. A method of voting for object centroids followed by a model fitting step was described in [20], but we assume having no CAD models for test objects in this paper. We would also have to consider 6DOF poses, complicating the approach considerably.

Whereas the segmentation of objects is not uniquely defined, there are still regularities in the number of parts they are broken up into. As shown in Fig. 4, the distribution of the number of different object parts, generated in the training stage of the part-graph-based hashing algorithm, can be modeled as a Poisson distribution, with an average error of 1.7% (and at most roughly 9%).

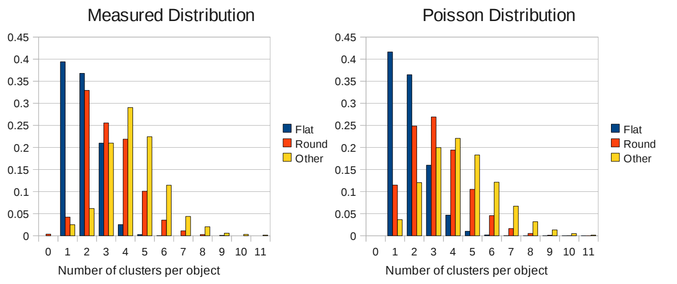

Fig. 4. Distribution of number of parts (see Fig. 3 column 3) per object and their approximation with a Poisson distribution.

The Poisson distribution described by Eq. 1 describes the probability of different number of events occurring in a given interval, which we interpret here as the number of part boundaries encountered over the surface of the scanned object. The parameter λ is the mean of number of parts, which in our case is 0.876 for flat, 2.166 for round, and 3.317 for other object types.

$$P(k\ parts\ forming\ a\ single\ object) = \lambda^k \exp -\lambda / k! \quad (1)$$

This simple model is used to judge if a group of parts of the same geometric category forms a single object or if the robot should try to interact with it. We cut the probabilities at 0.3 for flat and 0.15 for round objects.

Example: To demonstrate this, from the right part of Fig. 4 we can deduce that the flat object is most likely to consist of 1 or 2 parts. The test scene with 2 boxes (Fig. 3) was categorized as one object (column 4), but in column 3 we notice that there are 6 parts in the scene. The probability for 1 object consisting of 6 parts is below the 0.3 value according to the Poisson distribution and clearly indicates an over-segmentation error and the need for the robot to segment this region interactively.

IV. Push Contact Point Estimation

Once the over or under segmented region of interest has been identified according to the above generated distribution, the appropriate contact points between the objects in the scene and the robot's end effector must be determined. Furthermore, the direction the robot's end effector should move must be chosen.

In this paper we apply our previously developed approach based on the local concavities [8]. Since most commonly encountered household items have convex outlines when observed from above, our system uses local concavities in the 2D contour of an object group as an indicator for boundaries between the objects. The robot separates objects from each other by pushing its end effector in between these boundaries. As the implementation details of the corner-based pushing go beyond the scope of this paper, we refer the reader to [8] for details.

V. Textureless Object Segmentation

In this section we describe the selected RGBD features suitable for the tracking of textureless objects and the particle filtering-based tracking library. The features are estimated on the above classified list of object hypotheses O from the RGBD point cloud. RGB and the depth measurements in the point cloud are time synchronized and registered. We employ 3D circle and 3D cylinder point cloud features for the round objects and 3D line and 3D corner point cloud features for the flat objects. The rationale behind this selection of features is that they are all fast to compute and yet distinctive enough for tracking with the proposed tracking algorithm. The latter uses a combination of the visual appearance and the geometrical structure of the feature to compute the likelihood function of the feature hypothesis.

A. RGBD Features

In order to obtain a 3D line point cloud we first find object edge candidates in the cluttered scene using curvature values computed in the input point cloud from the Kinect sensor. Next we fit a line model to the object edge candidates using RANSAC [24] and finally pad the line with neighboring points on the object within a radius of $5cm$. 3D corner point clouds are determined using the 3D variant of the Harris corner detector as implemented in the Point Cloud Library (PCL)(pointclouds.org) and padded with neighboring points on the object within a radius of $5cm$ as well. Padding of both features is necessary in order to guarantee computation of a better likelihood function needed by the tracker as explained in the following subsection. The features are shown in Fig. 3 columns 2 and 5, 1st row.

To obtain a 3D cylinder point cloud, we also use a RANSAC model which is based on the fact that on a cylinder surface, all normals are both orthogonal to the cylinder axis and intersect it. We consider the two lines defined by two sample points and their corresponding normals as two skew lines, and the shortest connecting line segment as the axis. Determining the radius is then a matter of computing the distance of one of the sample points to the axis. By setting the cylinder axis perpendicular to the table results are more robust, but is not mandatory. Finally, the generation of the 3D circle is also done using RANSAC by projecting a sample point into the 3D circle's plane and computing the distance between this point and the point obtained as an intersection of the line from the circle's center with the circle's boundary, whereas the line is passing through the projected sample point. The features are shown in the 2nd row of Fig. 3 columns 2 and 5.

B. Particle Filtering-based Tracking of RGBD Pointclouds

The feature point clouds extracted above are then passed to the particle filter-based tracker as reference models. The tracker consists of four steps: i) the above described reference model selection, ii) pose update and re-sampling, iii) computation of the likelihood and iv) weight normalization. In the pose update step we use a ratio between a constant position and a constant velocity motion model which allows us to achieve efficient tracking with a lesser number of the particles. In the re-sampling phase we utilize Walkers Alias Method [25]. The likelihood function l_j of the hypotheses in the third step is computed as in Eq. 2 and is based on the similarity between the nearest points pair of the reference point (p_j) cloud and the input data (q_j). Similarity is defined as a product of a term describing the points pair's euclidean distance $l_{euclidean}$ and a term describing points pair's match in the HSV (Hue, Saturation, Value) color space l_{color}. α and β are the weight factors set to 0.5 in our case.

$$\begin{aligned} l_j &= l_{euclidean}(p_j, q_j) l_{color}(p_j, q_j) \\ l_{euclidean}(p_j, q_j) &= \frac{1}{1+\alpha|p_j - q_j|^2} \\ l_{color}(p_j, q_j) &= \frac{1}{1+\beta|p_{j,hsv} - q_{j,hsv}|^2} \end{aligned} \tag{2}$$

To obtain the model's weight we sum over likelihood values for every points pair in the reference model as follows: $w_i = \sum_j l_j$. This likelihood function assures a combined matching of model's structure and visual appearance. In the final step we normalize the previously computed model weight by applying a relative normalization as described in [26]. The real-time operation of the algorithm is made possible through various optimization techniques such as downsampling of the point clouds, openMP parallelization and KLD-based (Kullback-Leibler Divergence) sampling [27] to select the optimal number of particles.

Why not to track object parts? To answer this question we refer the reader to scene 1 in Fig. 3, column 3 where top surfaces of both boxes were grouped into one segment. Had we taken this segment as a reference cloud the tracking algorithm would fail due to its limitation to generate multiple reference clouds during tracking.

C. Trajectory Clustering

The tracked features' 3D trajectories (see Fig. 3 column 6) are clustered using Alg. 1 in order to find the feature-object associations. We treat each of the n RGBD features as a node in a graph, where edge weights represent the maximum number of consecutive violations of the relative distance variation threshold ($d_{threshold}$), i.e. *breaks* (optionally, also pose changes can be checked for better performance). The final connection matrix is obtained by removing the edges which have weights that exceed a given percentage ($p_{threshold}$) of the theoretic maximum number of frames. The distance between features which did not vary are then clustered together.

Algorithm 1: Graph-based trajectory clustering algorithm. A *break* between features means that the relative distance between them exceeded the given threshold.

```
/* number of tracked features $n$ and number of
   time steps $m$, relative distance variation
   threshold $d_{threshold}$, max allowed percent of
   consecutive breaks $p_{threshold}$, and the set of
   positions of each feature $T$                         */
Input: $n$, $m$, $d_{threshold}$, $p_{threshold}$, $T = \{t_1 ... t_m\}$
/* relative distances at $t_1$                           */
$D_{reference}$ = pairwiseL2($t_1$)
/* nr of consecutive breaks between features             */
$C_{breaks}$ = zeros($n$,$n$)
/* relative distances at $t_1$                           */
$T_{breaks}$ = zeros($m$,$n$,$n$)
/* count number of consecutive breaks                    */
foreach $t_i \in T$ do
    /* relative distances at $t_i$                       */
    $D_i$ = pairwiseL2($t_i$)
    /* deviation of distances                            */
    $E_i = |D_i - D_{reference}|$
    /* breaking feature pairs                            */
    $B_i = \{(f_1, f_2) | E_i[f_1, f_2] > d_{threshold}\}$
    foreach $(f_1, f_2) \in B_i$ do
        $C_{breaks}[f_1, f_2] + +$ /* increment counter  */
    foreach $(f_1, f_2) \notin B_i$ do
        $C_{breaks}[f_1, f_2] = 0$ /* reset counter      */
    $T_{breaks}[i]$ = $C_{breaks}$ /* save counter       */
/* maximum percentage of consecutive breaks              */
$M_{breaks}$ = max($T_{breaks}$)/$m$
/* final adjacency matrix                                */
$A$ = getConnections($M_{breaks} <= p_{threshold}$)
/* number of clusters based on Laplacian                 */
$nr_{clusters}$ = nrZeros(eigenValues(diag(degrees($A$)) - $A$))
/* get features clustered by connectivity                */
Output: $F_{clusters}$ = connectedComponents($A$)
```

Fig. 5 shows an evaluation of the clustering algorithm on 17 scenes from Fig. 7. The use of $p_{threshold}$ is clearly advantageous, and the method works well for a range of the $p_{threshold}$ and the $d_{threshold}$ parameters. Since too low values for $d_{threshold}$ over-segment the features, values over $1.5cm$ are used, and the possible under-segmentations solved by applying the whole method iteratively until all the objects are clearly separated.

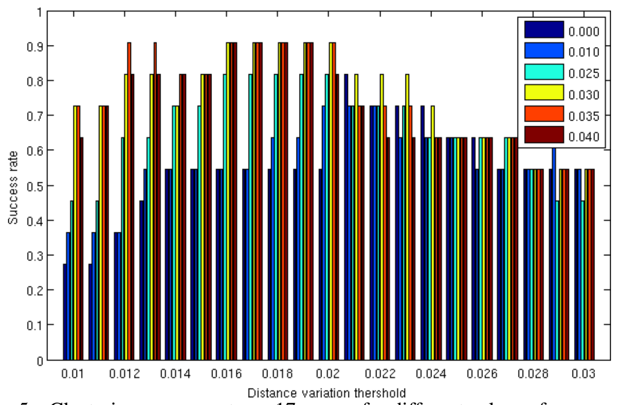

Fig. 5. Clustering success rate on 17 scenes for different values of $p_{threshold}$ (see legend) as a function of $d_{threshold}$ (in meters).

D. Dense Model Reconstruction

Considering the connected features $F_{clusters}$ as being part of the same object, we reconstruct the dense model of the object using region growing in normal space, which also makes use of the borders found at depth discontinuities, as shown in Alg. 2. The idea for the region growing constraints is based on the segmentation described by Mishra et al. [28], where the authors make use of a predefined fixation point and a border map. Since we already know the features that are part of the object, we can easily define a seed point for the region growing. In order to find the best possible seed point, we separate the connected features using euclidean clustering, calculate each of the resulting clusters' centroid, and then start growing from these. An important condition of the region growing is the assumption that objects are often composed of convex parts [29]. Therefore, we make sure that during region growing two points are assigned to the same region R_i if the angle eps_{thresh} between the vector connecting them and the points normal is close to obtuse (considering the sensor noise level [22], 89° were used). Once all region-feature pairs have been identified, we reconstruct the dense model. Since in the trajectory clustering step we already identified the features that belong to the same object, having multiple regions for the same object is easily dealt with by merging those regions for which the corresponding features belong to the same object into dense models R_j.

VI. Evaluation and Discussion

A. Experimental Setup

The system was evaluated on 17 scenes in different configurations as illustrated in Fig. 7. The scenes are numbered 1-17 and arranged according to the legend shown in Fig. 6. Though our system can iteratively cope with multi-object scenes, we performed the evaluation on two-object scenes with the finite number of scene configurations that can occur. These configurations can be split in three different ways, namely: i) size, ii) shape, and iii) arrangement. A scene may consist of two objects of different sizes or the same size. The objects may be either both flat or round or a combination of these two. They may also occur in different arrangements; completely separated, only touching, one on top of the other, or in solid contact. Solid contact refers to both objects being in contact with each other, whereby the contact area is larger than a single line (scene number 4 in Fig. 7). Some configurations are infeasible for our approach. For example a flat object and a round object cannot be of the same size, or round object on top of another round object cannot be pushed (one mug on top of another mug). It is also not possible to have a round object that is in solid contact with another round object. For this case we consider solid contact as being two objects touching with more than one line, for example in scene number 17 where also the handle of the mug touches the juice box.

Algorithm 2: Region growing with normals & boundaries.

```
/* set of features $F_{clusters}$, distance threshold
   $d_{roi\_thresh}$, angle threshold $eps_{thresh}$, seed queue
   $sq$, regions list $R$, current region $R_i$, list of
   processed points $processed$                              */
Input: $F_{clusters}$, $d_{roi\_thresh}$, $eps_{thresh}$
foreach $f_i \in F_{clusters}$ do
    $p_{s,i}$:= centroid($f_i$) $sq.add(p_{s,i})$ /* select a seed point
        and add it to a queue                                */
    $processed(p_{s,i})$ = $true$
    $R_i := \{p_{s,i}\}$ /* initialize region                */
    while $sq.notempty()$ do
        $N := \{q_j \| dist(q_j, R_i[c]) < d_{roi\_thresh}\}$
        /* select neighborhood                               */
        foreach $q_j \in N$ do
            if $processed(q_j)$ = $true$ then
                $continue$
            if $boundary(q_j)$ = $true$ then
                $stopgrowing = true$
                $R_i \leftarrow R_i \cup \{q_j\}$ $processed(q_j)$ = $true$
                $break$
            if $deg\ (\vec{p_{s,i}q_j}, norm(q_j)) > eps_{thresh}$ then
                $R_i \leftarrow R_i \cup \{q_j\}$
                $processed(q_j)$ = $true$
            else
                $break$
        if $stopgrowing = false$ && $\forall q_j \in N$ $boundary(q_j)$ =
        $false$ then
            $sq \leftarrow N$
    $R \leftarrow R_i$
foreach $R_i, R_j \in R$ do
    if $f_i f_j \in$ $same\ object$ then
        $R_i \leftarrow R_i \cup \{R_j\}$
Output: Dense models $R_j$
```

	arrangement	shape		
		flat-flat	round-round	round-flat
different size	separated	1	9	14
	touching	2	10	15
	on top	3	11	16
	in solid contact	4	-	17
same size	separated	5	12	-
	touching	6	13	-
	on top	7	-	-
	in solid contact	8	-	-

Fig. 6. Legend for the different scene configurations. The scenes are shown in Fig. 7.

It is important to emphasize that the above devised conventions refer to the scenes after a push. The scenes before interaction were designed such that it is difficult or impossible to segment them using static segmentation techniques.

Average time to segment one scene from Fig. 7 amounted to

$12.5s$ with the pre-segmentation taking $1.5s$, feature extraction $3.5s$, pushing $6s$ (tracking runs at $25fps$ for up to 10 features) and dense model reconstruction $1.5s$. Apart from tracking all modules perform linearly with the number of features and objects respectively and can thus easily be used for larger and more complex scenes. For all the scenes the push point estimation algorithm was used, the only exception being the 'on top' arrangements for which the algorithm does not generalize. For this reason and since the scope of the paper is on the priors from the static segmentation, RGBD features for textureless objects and the final dense model reconstruction, we performed the experiments by manually inducing motions into the corners of the scenes. In our future work we will address finding a generalized push point algorithm.

B. Results

All the experiments were performed three times for each of the 17 scenes. All the results are presented in Tab. I which shows the segmentation success rate for every scene. The corresponding figures for this data can be found in Fig. 7. The algorithm was never able to segment the scene number 8 and performed poorly for scenes 6 and 13. In these cases the contact surface of the two objects is large and the objects are of the same size. Erroneous reconstruction happens due to a lack of a sufficiently good boundary estimation near the touching surface, and therefore the region growing does not terminate. This could be alleviated by integrating texture/color-based segmentation methods, which we plan to investigate in the future.

It is important to note that the overall segmentation was successful in more than 82% of the experiments. Tab. II shows that the more objects differ and the less in contact they are the more successful the segmentation becomes. Our algorithm performs extremely well in the 'on top' arrangement which is very challenging for the static segmentation techniques.

We would like to draw the reader's attention to all the scenes with the round objects. It can be noted that the Kinect sensor from the used viewpoint (mounted on the head of the human size PR2 robot) always captures mugs as two spatially non-connected parts. In order to robustly merge these two parts using segmentation algorithms operating on point clouds or images of static scenes, model-based segmentation algorithms are required. While that constitutes a feasible solution, the system presented in this paper can easily deal with such scenes without a model by clustering the two parts of the mug since they move rigidly with respect to each other.

For the scene in bottom row of Fig. 3 we can observe that there is only one feature on the left object. All the clustering algorithms trying to explicitly cluster at least one pair of features with the constant relative distance over time would fail in this case. Using the graph-based clustering method we are able to disconnect the two nodes of the graph and infer that there is a single feature-object association.

VII. Conclusions and Future Work

We have presented a novel interactive segmentation system suitable for the segmentation of textureless objects in cluttered tabletop scenes. Integrated in the system are the static pre-segmentation based on geometrical categorization, a push point and direction estimation, RGBD features suitable for tracking of textureless objects, the graph-based trajectory clustering algorithm and the dense model reconstruction. A rigorous evaluation of the system on a set of 17 scenes showed successful segmentation in 82% of the cases. The results show the applicability of our system for objects of similar colors, shapes and sizes on predominantly flat and round surfaces.

Though the results of the presented system are very promising, there is still several improvements to be made. First, we will solve the problem depicted in scene number 8 by integrating color and the texture-based segmentation techniques. Second, we plan to improve our pushing heuristic such that we can deal with the 'on top' arrangement. This can be done by looking for 3D corners as pushing points. One could also use a different heuristic such as the singulation method presented by Chang et al. [15]. Lastly we will also address heavy occlusions and self-occlusions of RGBD features in the tracking step.

References

[1] D. Pangercic, M. Tenorth, D. Jain, and M. Beetz, "Combining Perception and Knowledge Processing for Everyday Manipulation," in *International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 18-22 2010, pp. 1065–1071.

[2] L. B. Cohen and C. H. Cashon, "Infant Perception and Cognition," in *Comprehensive Handbook of Psychology, Volume 6: Developmental Psychology*. Wiley and Sons, 2003, ch. II. Infancy, pp. 65–89.

[3] Z.-C. Marton, F. Balint-Benczedi, N. Blodow, L. C. Goron, and M. Beetz, "Object Categorization in Clutter using Additive Features and Hashing of Part-graph Descriptors," in *Spatial Cognition*, 2012.

[4] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[5] Q. Zhan, Y. Liang, and Y. Xiao, "Color-based segmentation of point clouds," 2009.

[6] D. Katz and O. Brock, "Interactive segmentation of articulated objects in 3d," in *Workshop on Mobile Manipulation at ICRA*, 2011.

[7] N. Bergström, C. H. Ek, M. Bjrkman, and D. Kragic, "Scene understanding through interactive perception," in *In 8th International Conference on Computer Vision Systems (ICVS)*, Sophia Antipolis, September 2011.

[8] C. Bersch, D. Pangercic, S. Osentoski, K. Hausman, Z.-C. Marton, R. Ueda, K. Okada, and M. Beetz, "Segmentation of cluttered scenes through interactive perception," in *RSS Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, Sydney, Australia, July 9–13 2012.

[9] Z. C. Marton, D. Pangercic, N. Blodow, and M. Beetz, "Combined 2D-3D Categorization and Classification for Multimodal Perception Systems," *The International Journal of Robotics Research*, 2011.

[10] J. Bruce, T. Balch, and M. M. Veloso, "Fast and inexpensive color image segmentation for interactive robots," in *International Conference on Intelligent Robots and Systems (IROS '00)*, vol. 3, October 2000, pp. 2061 – 2066.

[11] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, pp. 1124–1137, September 2004.

[12] P. Fitzpatrick, "First contact: an active vision approach to segmentation," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2003.

[13] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ser. ICRA'09, 2009.

[14] M. Gupta and G. S. Sukhatme, "Using manipulation primitives for brick sorting in clutter," *International Conference on Robotics and Automation (ICRA)*, 2012.

[15] L. Chang, J. R. Smith, and D. Fox, "Interactive singulation of objects from a pile," *International Conference on Robotics and Automation (ICRA)*, 2012.

Scene number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Success rate[%]	100	100	100	100	100	33,3	100	0	100	100	66,7	100	33,3	100	100	100	66,7

TABLE I
SEGMENTATION RESULTS FOR ALL 17 SCENES. FOR EACH SCENE THERE WERE 3 EXPERIMENTS CONDUCTED.

	all results	different size	same size	flat-flat	round-round	round-flat	apart	in contact (touching or in solid contact)	on top
Success rate[%]	82,4	93,9	61,1	79,2	80,0	91,7	100	66,7	91,7

TABLE II
SEGMENTATION SUCCESS RATES OF DIFFERENT SCENE CONFIGURATIONS.

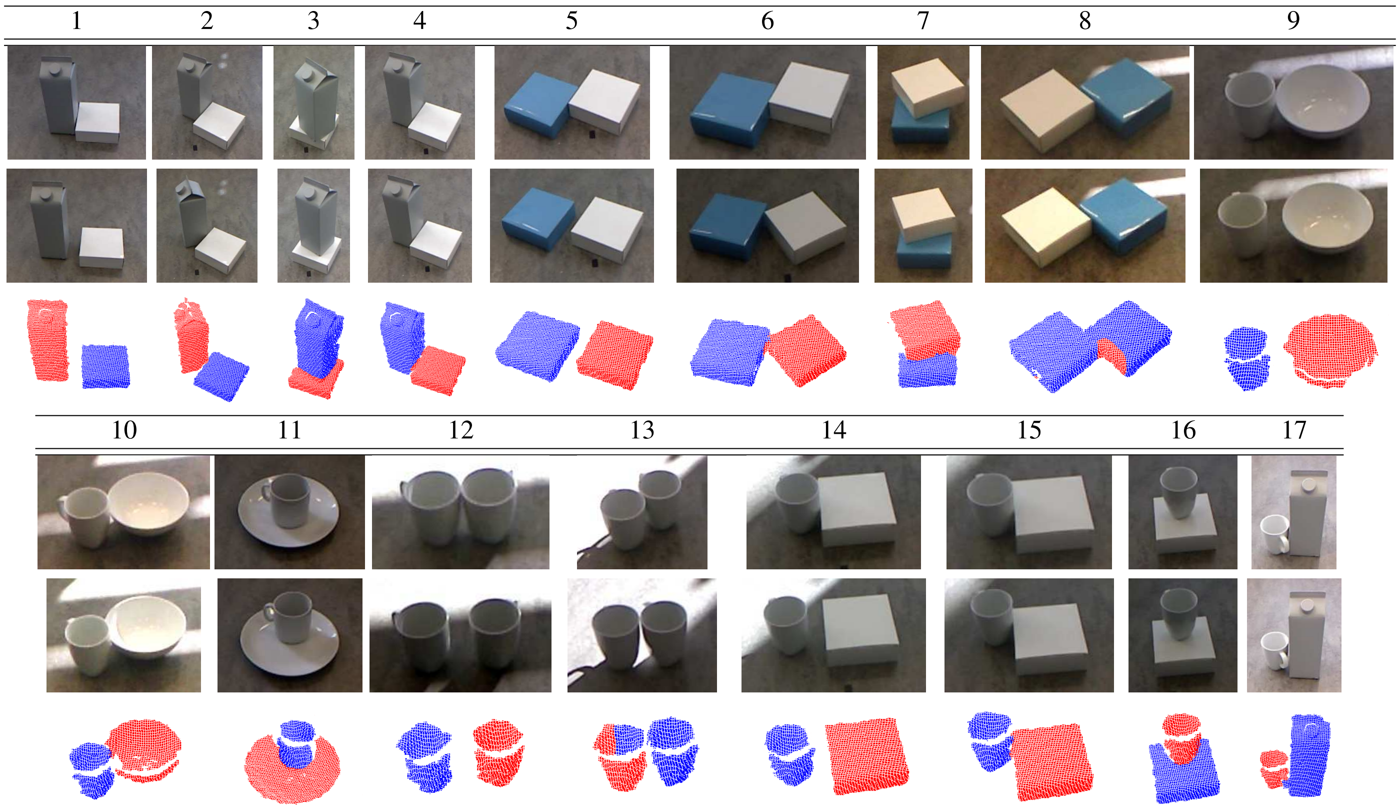

Fig. 7. Results of the segmentation for 17 scenes. 1st/4th image row: Image before the push for scenes. 2nd/5th: Image after the push for scenes. 3rd/6th: Point cloud after dense model reconstruction for scenes.

[16] H. Yang, T. Low, M. Cong, and A. Saxena, "Inferring 3d articulated models for box packaging robot," *CoRR*, vol. abs/1106.4632, 2011.

[17] J. Sturm, K. Konolige, C. Stachniss, and W. Burgard, "3D pose estimation, tracking and model learning of articulated objects from dense depth video using projected texture stereo," in *Advanced Reasoning with Depth Cameras at the Robotics: (RSS10)*, Zaragoze, Spain, June 2010.

[18] T. Malisiewicz and A. A. Efros, "Improving Spatial Support for Objects via Multiple Segmentations," in *Proceedings of the British Machine Vision Conference*, 2007.

[19] K. Lai and D. Fox, "Object recognition in 3d point clouds using web data and domain adaptation," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1019–1037, 2010.

[20] O. M. Mozos, Z. C. Marton, and M. Beetz, "Furniture Models Learned from the WWW – Using Web Catalogs to Locate and Categorize Unknown Furniture Pieces in 3D Laser Scans," *Robotics & Automation Magazine*, vol. 18, no. 2, pp. 22–32, 2011.

[21] A. Kanezaki, Z.-C. Marton, D. Pangercic, T. Harada, Y. Kuniyoshi, and M. Beetz, "Voxelized Shape and Color Histograms for RGB-D," in *ASP 11 at IROS 2011*, San Francisco, CA, USA, September, 25–30 2011.

[22] "Precision of the kinect sensor." [Online]. Available: http://www.ros.org/wiki/openni_kinect/kinect_accuracy

[23] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *International Conference on Robotics and Automation (ICRA)*, 2011.

[24] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[25] A. J. Walker, "An efficient method for generating discrete random variables with general distributions," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 253–256, Sept. 1977.

[26] P. Azad, D. Munch, T. Asfour, and R. Dillmann, "6-dof model-based tracking of arbitrarily shaped 3d objects," in *ICRA*, 2011.

[27] D. Fox, "Kld-sampling: Adaptive particle filters," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[28] A. K. Mishra and Y. Aloimonos, "Active segmentation with fixation," in *ICCV*, 2009.

[29] D. W. Jacobs, "Perceptual Organization As Generic Object Recognition," in *From Fragments to Objects - Segmentation and Grouping in Vision*, 2001, ch. IV. Models Of Segmentation And Grouping, pp. 295–329.