Segmentation of Cluttered Scenes of Textured Objects through Interactive Perception

Christian Bersch, Dejan Pangercic, Megha Gupta, Karol Hausman,
Zoltan-Csaba Marton, Andrei Haidu, Sarah Osentoski, Michael Beetz
{pangercic, hausman, marton, haidu, beetz}@cs.tum.edu,
meghagup@usc.edu, christian.bersch@googlemail.com, sarah.osentoski@us.bosch.com

Abstract— **This paper describes a novel object segmentation approach for autonomous service robots acting in human living environments. The proposed system allows a robot to effectively segment textured objects in cluttered scenes by leveraging its manipulation capabilities. In this interactive perception approach, 2D-features are tracked while the robot actively induces motions into a scene using its arm. The robot autonomously infers appropriate arm movements which can effectively separate objects. The resulting tracked feature trajectories are assigned to their corresponding object by using a novel clustering algorithm, which samples rigid motion hypotheses for the *a priori* unknown number of scene objects. We evaluated the approach on challenging scenes which included occluded objects, as well as objects of varying shapes and sizes.**

I. Introduction

A robot operating in human environments may be required to perform complex dexterous manipulation tasks in a variety of conditions. Service robots are likely to perform tasks that require them to interact with objects that populate human environments. For example, when emptying a shopping bag [1] the robot is likely to be confronted with a cluttered unstructured scene like the examples 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 can be difficult to distinguish between the object boundaries and texture patterns.

Similar to Katz et al. [2] and Bergstrom et al. [3], we propose a system that uses a robot arm to induce motions in a scene to enable effective object segmentation. Our system employs a combination of the following techniques: i) estimation of a contact point and a push direction of the robot's end effector by detecting the corners in the cluttered scene, ii) feature extraction using features proposed by Shi and Tomasi [4] and tracking using optical flow [5], and iii) a novel clustering algorithm to segment the objects.

Evaluation was performed on several simulated and real scenes with objects of different shapes and sizes. Using a PR2 robot [6], we show that our segmentation approach works very well on cluttered scenes, even in the presence of occluded objects and is fully sufficient for the robot to successfully grasp these objects. Our system is available as open source[1] and can be deployed on a robot equipped with a 2D-camera, a depth camera (or combination thereof) and at least one arm.

[1]http://ros.org/wiki/interactive_segmentation_textured

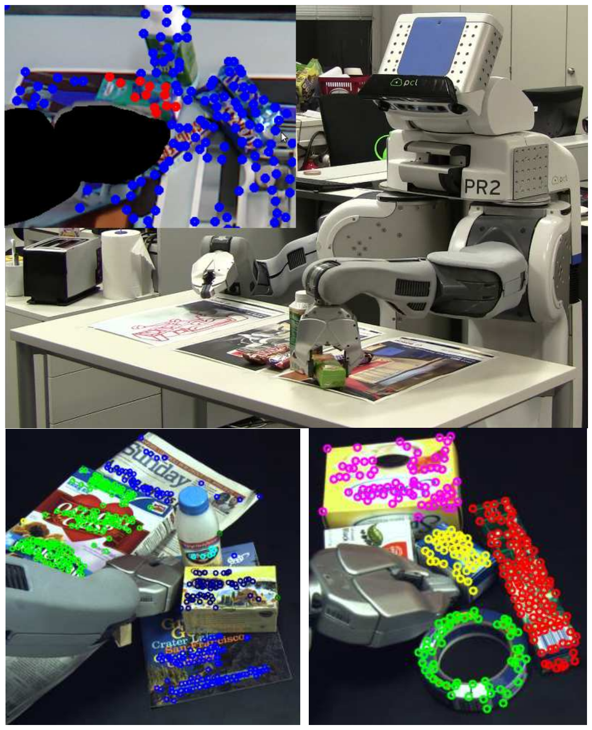


Fig. 1. Top: PR2 robot successfully picking-up the object after segmenting in it in clutter using herein proposed object segmentation algorithm. Bottom: the result of clustering of two highly cluttered scenes.

This paper provides the following main contributions:

- A novel clustering algorithm for 2D-feature trajectories (but not limited to) that is based on sampling rigid motion hypotheses for the *a priori* unknown number of scene objects;
- A heuristic for finding a contact point and a push direction for the robot's manipulator to induce distinct motions to effectively separate the objects.

The paper is organized as follows: the next section reviews related work, followed by a brief overview of the overall system design (Sec. III). Estimation of a contact point and a push direction is discussed in Sec. IV. Our algorithm for clustering feature trajectories is explained in Sec. V. Finally, we present our experimental results for several scenes in Sec. VI and conclude discussing potential future work.

Preprint submitted to 2013 IEEE International Conference on Robotics and Automation. Received September 17, 2012.

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 [7] or higher order features like in graph cut approaches [8]. Other approaches, such as that of Vidal et al. [9], employ camera motion for image segmentation.

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 [10] and Kenney et al. [11]. 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, we make use of the 3D data to segment the object candidates from the background and we use a novel clustering approach for the segmentation of textured objects.

Katz et al. [2] 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 a 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 [3] 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 seems to be the number of objects, which was never greater than two in the experimental results.

Some approaches examine how the perturbations can be planned to accumulate a sequence of motion cues. Gupta et al. [12] 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, this approach can not deal with heavily textured objects but could work well in combination with ours. Chang et al. [13] 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 and do not accumulate the transforms across the history of push actions.

The following works are dealing with the estimation of the articulation models for drawers, boxes, etc. [14], [15]. 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 Architecture

The overall system schema is depicted in Fig. 2 and consists of three main steps. In the first, a 3D point cloud of a static tabletop scene with household items is captured by a depth camera (in this case using the Kinect). Points belonging to the table are separated from points belonging to the group of objects. The shape of the group of objects is used to infer a contact point and a push direction for the robot's manipulator.

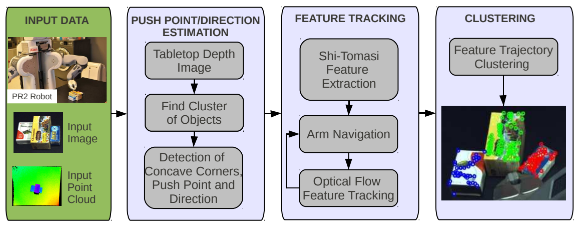

INPUT DATA
PR2 Robot
Input Image
Input Point Cloud
PUSH POINT/DIRECTION ESTIMATION
Tabletop Depth Image
Find Cluster of Objects
Detection of Concave Corners, Push Point and Direction
FEATURE TRACKING
Shi-Tomasi Feature Extraction
Arm Navigation
Optical Flow Feature Tracking
CLUSTERING
Feature Trajectory Clustering

Fig. 2. The system proposed in the paper consists of three main nodes: a node for estimating the initial contact point and the push direction, a node that extracts 2D-features and tracks them while it moves the robot arm in the push direction, and finally an object clustering node that assigns the tracked features to objects.

In the second step we capture a sequence of Kinect RGB camera images of the scene, while the robot pushes its end effector into the group of objects. Shi-Tomasi features are extracted from the first camera frame and then tracked over subsequent frames using optical flow. Once object motion is detected, the robot continues moving its end effector a predefined distance along the push direction. Since we assume the robot arm is calibrated and we have a model of its geometry, we employ a self-filter to exclude features from the robot's arm.

In the third step, the recorded feature trajectories are clustered and assigned to object hypotheses.

Preprint submitted to 2013 IEEE International Conference on Robotics and Automation. Received September 17, 2012.

IV. Estimation of Contact Point and Push Direction

To perform object segmentation based on the individual object motions induced by the robot, 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.

A. Where to Push - an Intuition

The contact points where the objects are pushed should be such that the push results in different objects moving differently resulting in different optical flows. The outcome of a push depends largely on the arrangement of objects but intuition suggests that pushing at corners of objects should result in not only translation but also rotation of the object directly being pushed. On the other hand, pushing at the sides of an object may result only in translation and any neighboring objects may also move together with it as a rigid body. Object corners are, therefore, good candidates for contact points for pushes irrespective of the object configuration. We use this heuristic to select our push points and later validate the accuracy of this heuristic through simulations in section VI-C.

B. Contact Points from Corners

We restrict the problem of finding a contact point to the table plane. Our algorithm employs 2D-image processing techniques to select contact point candidates. The table plane is estimated from the depth-camera's point cloud data using RANSAC [16] and separated from the object points. The remaining cloud points are projected into a virtual camera view above the table. In the next step we employ standard morphological operators and 2D-contour search [17] on projected cloud points to identify a closed region, R, corresponding to the group of objects.

This region's outer contour is then searched for strong local directional changes by applying a corner detector. As in the Shi-Tomasi corner detector [4], we compute the corner response for each pixel location $\mathbf{p} = (p_x, p_y)$ based on the covariance matrix $\mathbf{Z}$:

$$\mathbf{Z}(\mathbf{p}) = \left[\begin{array}{cc} \sum_{S(\mathbf{p})} I_x^2 & \sum_{S(\mathbf{p})} I_x I_y \\ \sum_{S(\mathbf{p})} I_x I_y & \sum_{S(\mathbf{p})} I_y^2 \end{array} \right] \quad , \tag{1}$$

where I_x and I_y are the image gradients in x and y direction and $S(\mathbf{p})$ the neighborhood of $\mathbf{p}$. A corner detector response is recorded if $min(\lambda_1, \lambda_2) > \theta$, where λ_1,λ_2 are the eigenvalues of $\mathbf{Z}$ and θ is a given threshold empirically set to 0.05. We also check for roughly equal eigenvalues, i.e. $\lambda_1 \approx \lambda_2$, ensuring that there are strong gradient responses in two approximately orthogonal directions. The local maxima of the smoothed corner responses, are the detected corners illustrated as circles in the middle row of Fig. 4. The smoothing was performed using a linear convolution with the $29x29$ Gaussian kernel which, together with the given choice of θ, is tolerant to the sensor noise[2] for the distances up to $1m$.

This method computes potential contact points for only one group of objects, which the robot wants to break up for segmentation. If the scene contains multiple object groups the method will be applied to each group separately.

C. Push Direction and Execution

The push direction at a corner is set to be parallel to the eigenvector corresponding to the larger eigenvalue of the covariance matrix $\mathbf{Z}(\mathbf{p})$ in Eq. 1. Intuitively, the dominant eigenvector will align with the dominant gradient direction. However, at a corner with two similar gradient responses in two directions, the eigenvector becomes the bisector. As only corners with roughly equal eigenvalues are chosen as potential contact point candidates, the eigenvector of each contact point candidate will bisect the angles of the contour at the corner location as shown in middle row of Fig. 4.

After determining the contact point candidates and the push directions in the 2D table plane, the end effector is moved within a constant small distance parallel to the table plane. A contact point is below the object's centroid and close to the friction vector between the object and the table, which avoids toppling over objects while pushing them. When there are multiple contact point candidates, the closest contact point to one of the end effectors and physically reachable by the robot arm, is selected.

V. Object Segmentation using Feature Trajectories

Once the robot's end effector touches the objects, the resulting object motions are used to discriminate between the different items on the table. Feature points are tracked in the scene and the resulting feature point trajectories are clustered. The clustering is based on the idea that features corresponding to the same objects must follow the same translations and rotations. The following assumptions with respect to objects were made:

- **Texture:** Each item has some texture over most of its surface, such that texture features can be used to appropriately represent an object for tracking.
- **Rigid Body:** Each item is a rigid body and not subject to deformations when interacting with the robot's end effector or other objects.

A. Feature Trajectory Generation using Optical Flow

We take advantage of the objects' texture properties by extracting $i = 1...N$ Shi-Tomasi features [4] at the pixel locations $\{\mathbf{p}_{i,0}\}_{i=1}^{N}$ from the initial scene at time $t = 0$, i.e. before an interaction with the robot took place. Feature locations are extracted using the same Shi-Tomasi feature detector as described in Sec. IV-B but in this case on the 2D textured image (Depicted in bottom row of Fig. 1) When the robot's end effector interacts with the object, a Lucas-Kanade tracker [18] is used to compute the optical flow of the sparse

[2]`http://www.ros.org/wiki/openni_kinect/kinect_accuracy`

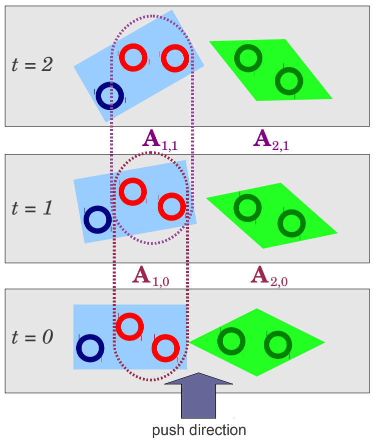


Fig. 3. Feature trajectory clustering with rigid motion hypotheses: Each feature i, depicted as a circle, is tracked over each time step t, forming a trajectory of feature positions S_i. After the robot finished its push motion, two features u and v, depicted as red circles, are randomly selected. From their trajectories S_u and S_v, a rigid transformation $\mathbf{A}_{k,t}$ is calculated that represents the rigid motion of u and v for each time increment from t to $t+1$. If u and v are on the same object, all other features will move according the sequence of rigid transformations $A_k = \{\mathbf{A}_{k,t}\}_{t=0}^{T-1}$, which serves as the rigid motion hypotheses for an object (e.g. the blue box). As the dark blue feature belongs to the same object as u and v, its motion can be explained by this motion hypothesis, and will thus be assigned to the same object. The motions of the dark green features located on a different object are poorly modeled by this motion hypothesis, and thus trigger the algorithm to create another motion hypothesis.

feature set. Using the optical flow, each feature's position $\mathbf{p}_{i,t}$ is recorded over the image frames at time $t = 0...T$ while the robot is interacting with the objects. That is, for each successfully tracked feature i, a trajectory $S_i = \{\mathbf{p}_{i,t}\}_{t=0}^{T}$ is obtained. The features are tracked with the average time resolution of $1.047s$ which is the average time needed to process one image frame.

B. Randomized Feature Trajectory Clustering with Rigid Motion Hypotheses

After calculating the set of all feature trajectories $\mathcal{S} \equiv \{S_i\}_{i=1}^{N}$, the goal is to partition this set such that all features belonging to the same object are assigned the same object index $c_i \in \{1,..,K\}$, where the number of objects K is not known *a priori*.

In other work on moving object segmentation, clustering has been applied directly to optical flow vectors [19], [20]. *However, in this context, where the robot induces the motion, the objects tend to be subject to strong rotational motions, which cause strongly non-collinear optical flow vectors*. Instead, we take advantage of the rigid body property of objects and assume that each subset of the features trajectories $\mathcal{S}$ belonging to the same object k are subjected to the same sequence of rigid transformation $A_k \equiv \{\mathbf{A}_{k,t}\}_{t=0}^{T-1}$, i.e. we cluster features with respect to how well rigid transformations can explain their motions. As the objects only move on the table plane, we restrict a possible rigid transformation $\mathbf{A}$ to be composed of a 2D-rotation $\mathbf{R}$, a 2D-translation $\mathbf{t}$ and a scaling component s, i.e. $\mathbf{A} = s \cdot [\mathbf{R}|\mathbf{t}]$. The scaling component compensates for the changes in size of the projected objects in the camera image. The actual scaling is not linear due to the perspective view, however, the error resulting from this linearization is small as the objects are displaced only in small amounts.

The clustering algorithm we propose is outlined in Alg. 1, and combines a divisive clustering approach with RANSAC-style model hypothesis sampling. At the core of the algorithm (lines 4–12, see also Fig. 3), we randomly draw 2 tracked features u,v and estimate a sequence of rigid transformations $A_{1,t}$ from their optical flow motions as first model hypothesis. The feature trajectories S_i that can be explained well by $A_{1,t}$ are considered "model inliers" and are removed from set of feature trajectories. From the remaining set, again 2 features are drawn to create a second model hypothesis $A_{2,t}$ and all inliers are removed. This process repeats until there are not enough features left to create a new model hypothesis. This process results in K hypotheses.

Algorithm 1: Randomized feature trajectory clustering. Mind that for the sake of clarity we do not write out the subscript m in the text explaining this algorithm.

```
1  Input: Set of feature trajectories $\mathcal{S} \equiv \{S_i\}_{i=1}^{N}$ where
      $S_i = \{\mathbf{p}_{i,t}\}_{t=0}^{T}$
2  Output: object cluster count $K$, object cluster
      assignments $\mathbf{c} = [c_i]_{i=1}^{N}$ where $c_i \in \{1,..,K\}$
3  for $m := 1$ to $M$ do
4      $k_m := 1$, $\mathcal{S}_m := \mathcal{S}$
5      while $|\mathcal{S}_m| \geq 2$ do
6          draw 2 random trajectories $S_u, S_v \in \mathcal{S}_m$
7          generate sequence of rigid transformations:
           $A_{k_m} \equiv \{\mathbf{A}_{k_m,t}\}_{t=0}^{T-1}$ from $(S_u, S_v)$
8          for $S_j$ in $\mathcal{S}_m$ do
9              sum squared residuals w.r.t to $A_{k_m}$:
               $r_{k_m,j} := \sum_{t=0}^{T-1} \|\mathbf{p}_{j,t+1} - \mathbf{A}_{k_m,t}\mathbf{p}_{j,t}\|_2^2$
10             if $r_{k_m,j} < THRESHOLD$ then
11                 $\mathcal{S}_m := \mathcal{S}_m \setminus \{S_j\}$
12         $k_m := k_m + 1$
13     $K_m := k_m$
14     for $S_i$ in $\mathcal{S}$ do
15         Assign each trajectory to best matching rigid
           transformation sequence:
           $c^*_{m,i} :=$ argmin$_{\{1,..,k_m,..,K_m-1\}}\, r_{k_m,i}$, where
           $r_{k_m,i} := \sum_{t=0}^{T-1} \|\mathbf{p}_{i,t+1} - \mathbf{A}_{k_m,t}\mathbf{p}_{i,t}\|_2^2$
16 Select best overall matching set of rigid transform
      sequences: $m^* :=$ argmin$_m \sum_{k_m=1}^{K_m} \frac{\sum_i r_{k_m,i} \cdot \mathbf{1}_{[c^*_{m,i}=k_m]}}{\sum_i \mathbf{1}_{[c^*_{m,i}=k_m]}}$
17 Return: $K := K_{m^*}$, $\mathbf{c} := \left[c^*_{m^*,i}\right]_{i=1}^{N}$
```

We bias the sampling of the 2 points (line 6) such that drawn feature pairs are not likely to be further apart than the object size $OS = 0.1m$ that the robot can grasp. For this, the first feature u is chosen uniformly and the probability p for choosing a feature i as the second point is proportional to the normalized Gaussian density function of the distance between $\mathbf{p}_i$ and $\mathbf{p}_u$:

$$p(i) \propto \exp\left(-\frac{\| \mathbf{p}_i - \mathbf{p}_u \|_2^2}{2\sigma^2}\right), \tag{2}$$

where σ is set to half of the object size OS in image space.

In line 7, a rigid transformation $\mathbf{A}_{k,t}$ is computed from the trajectories S_u and S_v at each time increment from t to $t+1$. A 4-DOF transformation $\mathbf{A}_{k,t}$ can be computed directly using geometrical considerations outlined in [21] from the two 2D-feature point locations $\mathbf{p}_u$ and $\mathbf{p}_v$ at t and $t+1$, such that:

$$\mathbf{p}_{u,t+1} = \mathbf{A}_{k,t}\mathbf{p}_{u,t} \quad \text{and} \quad \mathbf{p}_{v,t+1} = \mathbf{A}_{k,t}\mathbf{p}_{v,t} \quad . \tag{3}$$

We use the sum of squared residuals over all time increments, $r_{k,j} = \sum_{t=0}^{T-1} \|\mathbf{p}_{i,t+1} - \mathbf{A}_{k,t}\mathbf{p}_{i,t}\|_2^2$, as a measure of how well a feature trajectory S_i fits a transformation sequence A_k, where each residual is the difference vector between the actual feature location $\mathbf{p}_{i,t+1}$ and $\mathbf{A}_{k,t}\mathbf{p}_{i,t}$, the feature location predicted by $\mathbf{A}_{k,t}$. This measure is used to discriminate between inliers and outliers for the model generation process (line 10), as well as for the best assignment c_i^* of a trajectory to a model hypothesis (line 15). Note that the final assignment may not necessarily correspond to the model hypothesis for which the trajectory was an inlier if a later generated hypothesis explains its motions better. Furthermore, using a model to predict the features' motions at each time step, as compared to considering only the total movement induced by the robot, is effective at discriminating between objects that start or stop moving at different time steps during the robot interaction.

As each trajectory pair that is used for the model hypothesis generation is chosen in a randomized fashion, it can happen that a pair of features is chosen such that they are not on the same object. This can cause an erroneous partitioning process of the feature trajectory set, resulting in wrong model hypotheses. However, this problem can be overcome with a high probability by sampling from the whole hypotheses generation process M-times, where each set of model hypotheses is indexed by the iteration m in Alg. 1. This is explained in detail in Sec. V-C. We choose the best m according to the score function (line 16), which is the sum of summed squared residuals between each trajectory and its best matching model hypothesis, normalized by the number of feature trajectories assigned to this hypothesis. This measure thus favors sets of hypotheses that predict a smaller number of objects and where the number of features per object is roughly equal. Often erroneous hypotheses are only supported by few outlier features and are suppressed.

Even though we used the clustering algorithm for the clustering of the 2D features, the algorithm scales to 3D space as well.

C. Trajectory Clustering Complexity Analysis

Instead of sampling M-times the trajectory model generation process, one could generate a model hypothesis for all $\binom{N}{2} \approx \frac{N^2}{2}$ possible feature pairs (N - number of all extracted features as above), as done in quality-threshold clustering [22]. However, for a small maximal number of objects K, M can be small such that the computational complexity of our algorithm is much lower as shown in the following.

The probability that a draw of a pair of trajectories $\mathcal{S}_u$, $\mathcal{S}_v$ (line 7) is the result of the motion of the same object can be approximated if we neglect the bias in Eq. (2) and instead assume uniformly random draws from all detected feature trajectories. Given the true number of objects K and the number of feature trajectories N_k on an object $k \in \{1, ..., K\} := \mathcal{K}$, the probability to select any 2 feature trajectories from the same object in the first draw $w = 0$ is

$$P_{k,0}(2 \text{ traj. on object } k) = \frac{N_k \cdot (N_k - 1)}{N \cdot (N-1)} \quad . \tag{4}$$

Thus the probability to have 2 feature trajectories selected on *any* of the K objects together in the first draw $w = 0$ is:

$$P_0 = \sum_{k \in \mathcal{K}} \frac{N_k \cdot (N_k - 1)}{N \cdot (N-1)} \approx \sum_{k \in \mathcal{K}} \left(\frac{N_k}{N}\right)^2, \quad N \gg 1 \quad . \tag{5}$$

If we assume that N_k is the same for all objects, i.e. $N_k := \bar{N}_k = \frac{N}{K}$, this simplifies to $P_0 = \frac{1}{K}$.

Once a draw is made, all trajectories matching the model hypothesis defined by the draw are assigned to that model and are removed from $\mathcal{S}_m$. The next draw $w = 1$ is made from the remaining $N - N_k$ trajectories of the remaining $K - 1$ objects, given that the trajectory assignment (line 10-11) discriminates sufficiently between the trajectories belonging to that model and those that do not. Analogously the probability for drawing two feature trajectories from any of the remaining objects in draw w, conditioned that all previous draws were correct, is:

$$P_w = \sum_{k \in \mathcal{K} \setminus \bar{\mathcal{K}}} \left(\frac{N_k}{N - \sum_{k' \in \bar{\mathcal{K}}} N_{k'}}\right)^2 \overset{N_k = \bar{N}_k}{=\!=} \frac{1}{K - w} \quad , \tag{6}$$

where $\bar{\mathcal{K}}$ is the set of modeled objects whose trajectories have been removed from $\mathcal{S}_m$.

The probability that for all draws the drawn feature trajectory pairs are together on an object thus is:

$$P = \prod_{w=0}^{K-1} P_w = \frac{1}{K!} \quad . \tag{7}$$

The probability P directly corresponds to the inlier probability in RANSAC. Thus, one can similarly estimate the number of times M that above drawing process should be executed in order to find a set of correct motion model hypotheses with a given probability $\alpha < 1$. That is, the algorithm returns a good segmentation with probability α. As $1 - \alpha = (1 - P)^M$ is the probability that in none out of M drawing processes all K drawn trajectory pairs were together on an object, M can be calculated as:

$$M = \frac{\log(1-\alpha)}{\log(1-P)} = \frac{\log(1-\alpha)}{\log(1 - \frac{1}{K!})} \quad . \tag{8}$$

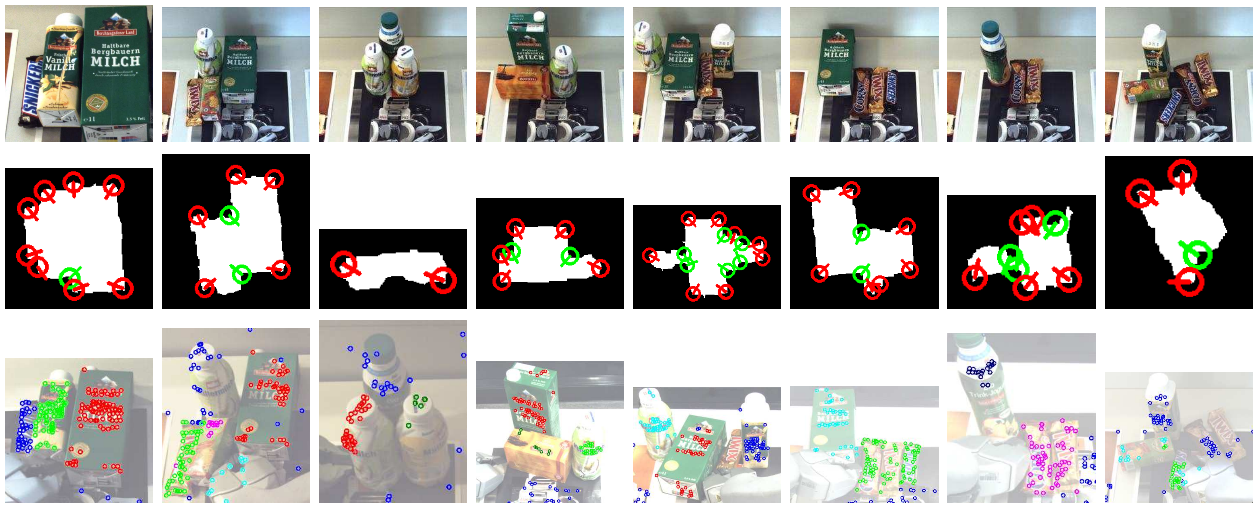


Fig. 4. Test scenes 1 to 8 from left to right. Top row: original scenes, middle row: contact point estimation, bottom row: segmentation after the first push cycle. Please note, that the 2D contours in the middle row are generated from the virtual view above the table which may slightly change the perspective. Features on the objects that did not get pushed are in the same cluster (denoted by the same color) as background features.

For example for $K = 4$ objects and $\alpha = 0.95$, $M = 72$ sampling runs are required.

However, in our experiments, $M = 20$ proved to be sufficient, since we bias the drawing of trajectories towards features that are initially close to each other, such that the probability P_w is higher than in random sampling. It is important to notice that this bias as controlled by σ in Eq. (2) may not be too strong, as close-by feature pairs decrease the accuracy when estimating a motion model for an object from their trajectories. This is because the feature displacements are tracked only with pixel precision such that the rotation estimate of close-by feature pairs is subject to larger noise, which can result in splitting an object in the segmentation.

VI. Experiments

A. Experimental Setup

We evaluated the proposed approach on simulated and real scenes. For the latter, our system was deployed on Willow Garage's PR2 robot. Depth and RGB images were taken from a Kinect sensor mounted on the robot's head (See Fig. 1).

B. Real Scenes

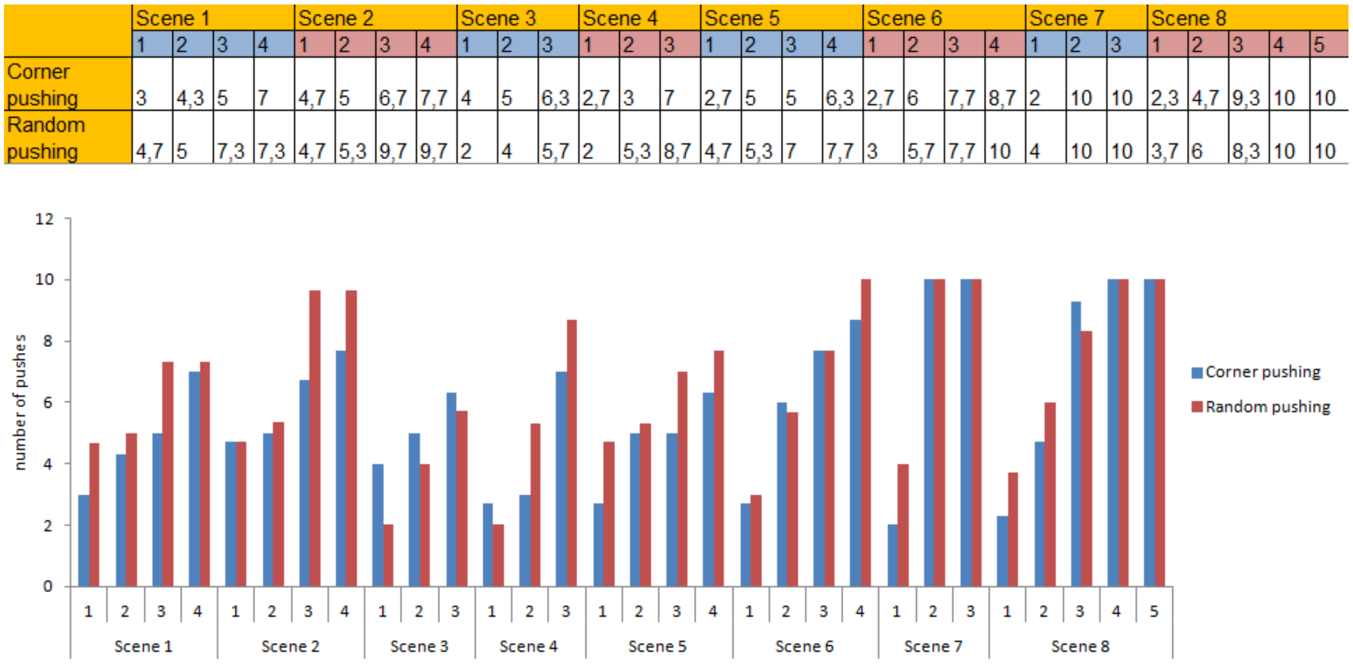
We arranged eight tabletop scenes with the cluttered background shown in Fig. 4. and evaluated the segmentation success rate given corner-based pushing and random pushing. In the latter mode we randomly sample poses from the set of reachable poses along the object pile contour. In the corner-based pushing experiments we also evaluated the correctness of detected contact points and push directions. After contact point and direction initialization, the robot entered into a push-cluster cycle until one or more objects were successfully segmented (the number of pushes was not exclusive to single object) or we reached a maximum 10 number of pushes. For every push the robot's end-effector travelled for $1cm$ along the push direction.

For each of the scenes we carried out three segmentation runs and present the average results for in Fig. 5. In every run the success of the segmentation was inspected visually by the human operator and the objects were removed from the scene by the operator upon the successful segmentation. In all 24 runs for the corner-based pushing all the contact points and the push directions were successfully chosen and executed. As shown in Fig. 5 an informative, corner-based pushing results in a faster and more reliable segmentation of objects. Using random pushing, there were 10 runs in which the robot failed to segment the whole scene (total of 26 unsegmented objects). For the corner-based pushing we only had 3 unsegmented scenes and total of 10 non-segmented objects, while exerting 0.6 pushes less on average as in the case of random pushing. Across all runs using corner-based pushing 89% of all objects were segmented successfully. The most frequent source of failures for the random pushing is depicted in the right part of Fig. 7 where the push motion was induced such that all objects moved rigidly with respect to each other. In terms of average computational costs per scene, corner detection and direction estimation took $0.2s$, arm to corner movement $5s$ and Shi-Tomasi feature extraction $0.1s$. Every next push took $0.3s$ and the clustering of every next image frame requires $1.047s$. The total average time to segment 1 scene was $\sim$1 minute.

C. Simulations

We carried out several simulations in physics-based simulator Gazebo[3] to validate our corner pushing heuristic. To verify that pushing at corners is indeed more effective, we spawned different scenes in Gazebo with two or three closely placed objects. Objects were flat and round, in different orientations and arranged such that they were in solid contact or in single point contact. We then simulated pushing at these objects with a PR2 gripper at many different contact points along the bounding box of the objects and in many different directions. More precisely, we chose points along

[3]`http://gazebosim.org/`



	Scene 1				Scene 2				Scene 3			Scene 4			Scene 5				Scene 6				Scene 7			Scene 8				
	1	2	3	4	1	2	3	4	1	2	3	1	2	3	1	2	3	4	1	2	3	4	1	2	3	1	2	3	4	5
Corner pushing	3	4,3	5	7	4,7	5	6,7	7,7	4	5	6,3	2,7	3	7	2,7	5	5	6,3	2,7	6	7,7	8,7	2	10	10	2,3	4,7	9,3	10	10
Random pushing	4,7	5	7,3	7,3	4,7	5,3	9,7	9,7	2	4	5,7	2	5,3	8,7	4,7	5,3	7	7,7	3	5,7	7,7	10	4	10	10	3,7	6	8,3	10	10

Fig. 5. Results of the segmentation of objects depicted in Fig. 4 using random vs. corner-based pushing. The tabular (upper) part of the figure denotes the average number of pushes over 3 runs needed to segment the respective object in the respective scene. Number 10 (maximum number of pushes allowed) means the robot failed to segment the object. The same statistics is also depicted as a bar chart in the bottom part of the Figure for clarity. X-axis represents the scene and the object number, Y-axis the number of pushes.

the bounding box spaced $2cm$ apart and for every such point, the gripper simulated a sequence of 2 pushes in 7 different directions $15°$ apart (See Fig. 6). The starting gripper pose and the object poses before and after every push were recorded. Then Shi-Tomasi features with known but randomly determined locations were spawned artificially on the objects. Based on the recorded object poses, the locations of all the features were computed after every push. This enabled us to compute the simulated optical flows. The feature trajectories so obtained were then clustered using Algorithm 1. The contact points for the pushes which resulted in a successful segmentation of objects were then observed and the number of successful corner pushes were counted. A push was classified as a corner push if the contact point is less than $1cm$ away from an object corner. We carried out 5 runs[4] on every of 24 different scenes (11 scenes with 2 objects, 13 scenes with 3 objects), which resulted in an average of 381.5 push sequences for a scene out of which an average of 14.9 pushes were successful in segmenting all the objects in the scene correctly. Out of these, 7.25 pushes were corner pushes. There were an average of 10 object corners in each scene. From this it follows that there were on average 70 corner pushes and 311.5 other pushes. This gives the segmentation success of 10% for the corner pushes and 2.4% for the non-corner ones. The reason for the low overall segmentation result compared to the real scenes above is on the one hand in that the scenes in the simulation included the single contact points between the objects and on the other hand in that various non-favorable orientations per contact point were computed and executed. We observed that corner pushing was successful in all the scenes while side pushing was successful only when the objects were in single point contact. When the objects were next to each other and similar in size, pushing at the sides resulted in the objects moving together as a single rigid body, thus making the algorithm fail. In such cases, only corner pushes succeeded. These simulations thus prove the benefits of corner pushing irrespective of object arrangement.

[4]Please mind that since Gazebo uses an ODE engine which is based on linear complementarity problem constraint formulation and since the simulation is defendant on the CPU load, the runs are not fully deterministic.

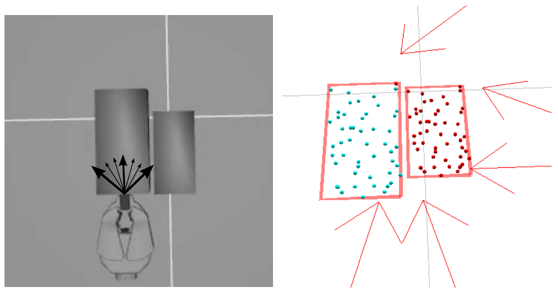


Fig. 6. Screenshots from the Gazebo simulation of a two object scene (left) and the corresponding visualization of the segmentation result. The black arrows in the left image show the 7 push directions for a single contact point. The dots on the objects in the right image represent features and their colors represent the cluster they were assigned to for a particular successful push sequence. The red arrows represent the starting gripper positions and directions of all the successful push sequences in a simulation run.

D. Grasping

We ran a grasping experiment on the scene 8 (Fig.4) and use an associated point cloud for the calculation of the object pose. To compute the latter we take the set of 3D points corresponding to the the set of 2D features from the successfully segmented cluster and apply an Euclidean clustering to remove possible outliers. We then compute the 3D centroid to obtain the object position and then use the Principle Component Analysis to compute the orientation. The result of this experiment is presented a video accompanying this submission. For high resolution please refer to: `http://youtu.be/4VVov6E3iiM`.

E. Discussion

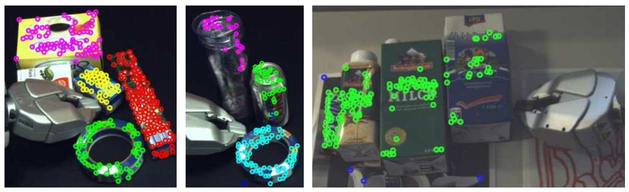


Fig. 7. Failure cases.

To exemplify some of the failed cases we would like to direct reader's attention to Fig. 7. One failure was caused by unsuccessful tracking of the features through the image sequence, for instance when the robot's arm occluded initially detected features (e.g. pepper box in the left scene). A similar effect was observed, when an object was rotated due to the robot interaction and features on its vertical surface were occluded by the object itself. In the middle scene, a semi-transparent and a reflective object was used. The failure was caused because the features were lost during tracking or they moved entirely inconsistently as the reflection pattern changed. In the right scene the objects moved together as a single rigid body as a result of the random pushing.

VII. Conclusions and Future Work

We presented an interactive perception system suitable for the segmentation of objects in cluttered scenes for the purpose of mobile manipulation. Integrated in the system are two novel algorithmic contributions: randomized clustering of 2D-feature trajectories based on sampling rigid motion hypotheses and the estimation of a contact point and a push direction for the robot's manipulator to induce distinct motions that enable the clustering. We evaluated the system on a set of challenging cluttered simulated and real scenes and successfully segmented the latter in more than 89% of cases. The results show applicability of our system for objects of various sizes, shapes and surface. Future work includes integrating our approach with other object segmentation techniques discussed in the related work in order to account for textureless objects and to further improve the segmentation rate. We also plan to integrate an arm motion and grasp planner [23] which will enable the robot to perform robust grasping and deal with more complex scenes.

References

[1] E. Klingbeil, D. Drao, B. Carpenter, V. Ganapathi, O. Khatib, and A. Y. Ng., "Grasping with application to an autonomous checkout robot," in *In International Conference on Robotics and Automation (ICRA)*, 2011.

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

[3] 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.

[4] J. Shi and C. Tomasi, "Good features to track," in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, 1994, pp. 593 – 600.

[5] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, April 1981, pp. 674–679.

[6] Willow Garage, "Personal Robot 2 (PR2)." [Online]. Available: www.willowgarage.com

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

[8] 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.

[9] R. Vidal, Y. Ma, and S. Soatto, "Two-view multibody structure from motion," *International Journal of Computer Vision*, vol. 68, p. 2006, 2004.

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

[11] 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.

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

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

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

[15] 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.

[16] 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.

[17] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[18] J. Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the algorithm," Jean-YvesBouguet, 2002.

[19] J. Klappstein, T. Vaudrey, C. Rabe, A. Wedel, and R. Klette, "Moving object segmentation using optical flow and depth information," in *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*, ser. PSIVT '09. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 611–623.

[20] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *Proceedings of the 11th European conference on Computer vision: Part V*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 282–295.

[21] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[22] L. J. Heyer, S. Kruglyak, and S. Yooseph, "Exploring Expression Data: Identification and Analysis of Coexpressed Genes," *Genome Research*, vol. 9, no. 11, pp. 1106–1115, Nov. 1999.

[23] M. Dogar and S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, October 2010.

Preprint submitted to 2013 IEEE International Conference on Robotics and Automation. Received September 17, 2012.