Segmentation of Cluttered Scenes through Interactive Perception

Christian Bersch, Dejan Pangercic, Sarah Osentoski, Karol Hausman, Zoltan-Csaba Marton, Ryohei Ueda, Kei Okada, Michael Beetz
{pangercic, hausman, marton, beetz}@cs.tum.edu, {ueda, k-okada}@jsk.t.u-tokyo.ac.jp,
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. Finally, we also briefly describe and discuss our most recent work towards the segmentation of textureless objects.**

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 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. [1] and Bergstrom et al. [2], 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 concave corners in the cluttered scene, ii) feature extraction using features proposed by Shi and Tomasi [3] and tracking using optical flow [4], and iii) a novel clustering algorithm to segment the objects.

Evaluation was performed on several scenes with objects of different shapes and sizes. Using a PR2 robot, 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 opensource[1] and can be deployed on a robot equipped with a 2D-camera, a depth camera and at least one arm.

This paper provides the following main contributions:

[1]http://www.ros.org/wiki/pr2_interactive_segmentation

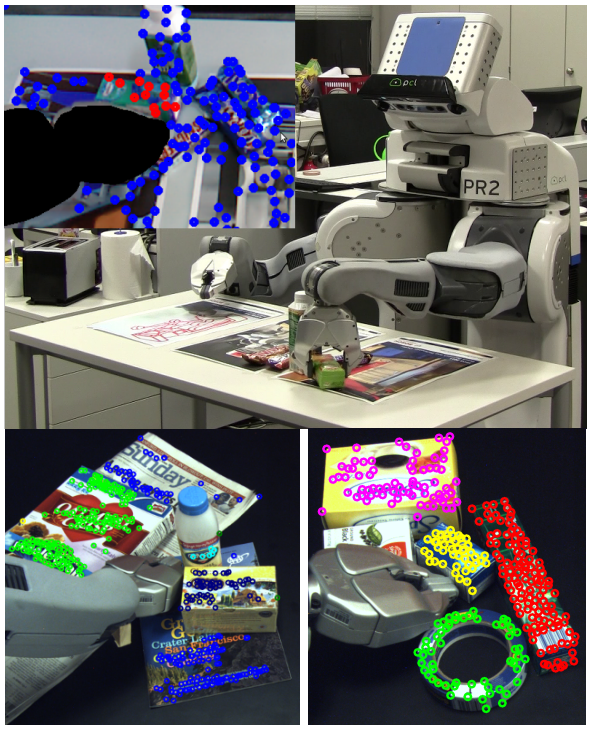

Fig. 1. Top: PR2 robot successfully picking-up the object after segmenting in it in clutter using herein proposed object segmentation algorithm. Bottom: the result of clustering of two highly cluttered scenes.

- A novel clustering algorithm for 2D-feature trajectories 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.

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 [5] or higher order features like in graph cut approaches [6].

This paper focuses on interactive scene segmentation by adding robotic arm manipulation into the perception loop.

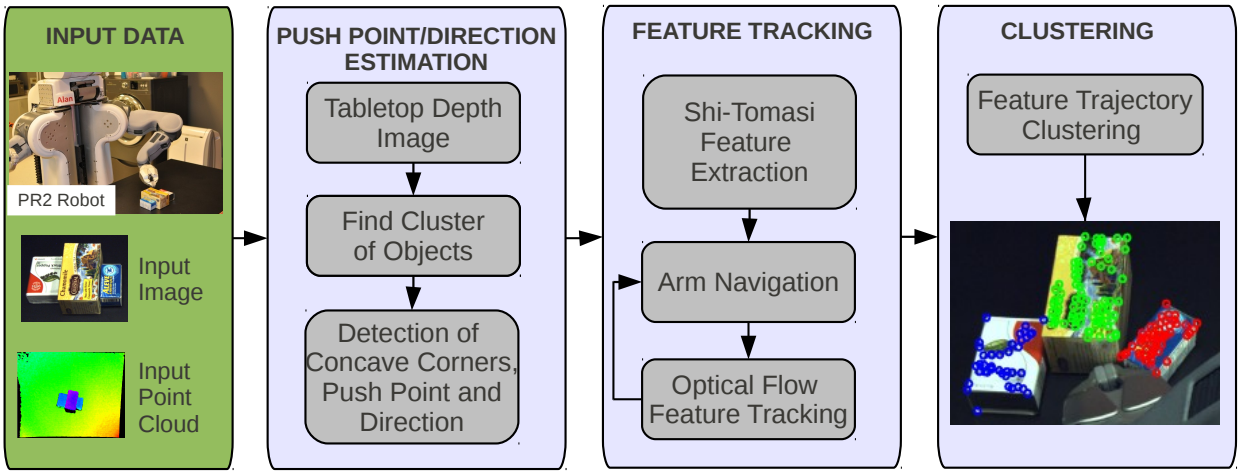

Fig. 2. The system proposed in the paper consists of three main nodes: a node for estimating the initial contact point and the push direction, a node that extracts 2D-features and tracks them while it moves the robot arm in the push direction, and finally an object clustering node that assigns the tracked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

Segmentation of rigid objects from a video stream of objects being moved by the robot has been addressed by Fitzpatrick [7] and Kenney et al. [8]. 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. [1] 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 [2] 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. [9] 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. [10] 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.

III. System Architecture

The overall system schema is depicted in Fig. 2 and consists of three main nodes. 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, as detailed in Sec. IV.

The node takes a sequence of 5-megapixel resolution 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 on subsequent frames using optical flow. Once object motion

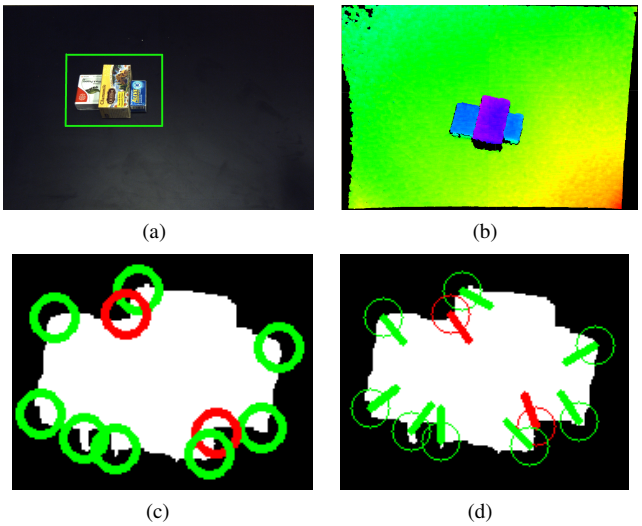

Fig. 3. Estimation of the contact point and the push direction. Top-left figure: original scene. Top-right figure: depth image as seen from the virtual camera positioned above the table. Bottom-left figure: Extracted contour of the object cluster, convex corners are shown in green, concave corners in red. Bottom-right figure: Direction of the dominant eigenvectors at the corners.

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 node, the recorded feature trajectories are clustered and assigned to object hypotheses, using the algorithm described in Sec. V.

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.

In this work, we consider a cluttered tabletop scene. 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. In the following, we describe a heuristic to determine a contact point and a push direction from depth-sensor data.

A. Contact Points from Concave 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 [11] and separated from the object points. The remaining cloud points are projected into a virtual camera view above the table. Since the projected cloud points are sparse, we employ standard morphological operators and 2D-contour search [12] to identify a closed region, R, corresponding to the group of objects. These steps are shown in Fig. 3.

This region's outer contour is then searched for strong local directional changes by applying a corner detector and subsequently the corners that are placed at local concavities are selected. As in the Shi-Tomasi corner detector [3], we compute the corner response for each pixel location $\mathbf{p} = (p_x, p_y)$ based on the covariance matrix $\mathbf{Z}$:

$$\mathbf{Z}(\mathbf{p}) = \begin{bmatrix} \sum_{S(\mathbf{p})} I_x^2 & \sum_{S(\mathbf{p})} I_x I_y \\ \sum_{S(\mathbf{p})} I_x I_y & \sum_{S(\mathbf{p})} I_y^2 \end{bmatrix} , \qquad (1)$$

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. 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 Fig. 3(c).

The concavity of each corner is estimated using a small circular neighborhood. If a larger portion of this neighborhood is inside R rather than outside, the corner must be a concave part of R's contour and is shown red in Fig. 3(c). This method effectively handles noise in terms of directional changes. Only the concave corners are considered contact point candidates, unless no corner is found fulfilling above concavity criterion. 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.

B. 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 Fig. 3(d).

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 an object's center of gravity 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. Textured Object Segmentation using Feature Trajectories

Once the robot's end effector touches the objects, the resulting object motions are used to discriminate between

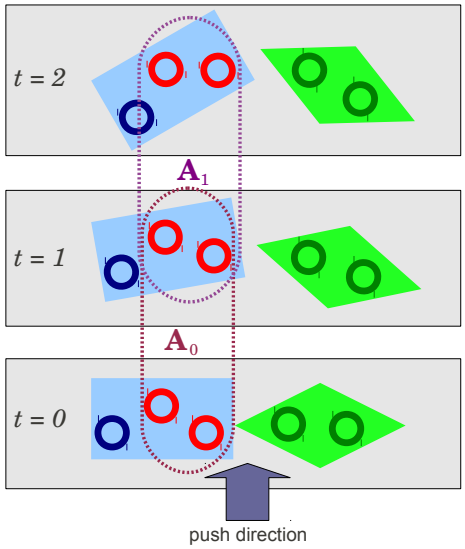

Fig. 4. Feature trajectory clustering with rigid motion hypotheses: Each feature i, depicted as a circle, is tracked over each time step t, forming a trajectory of feature positions S_i. After the robot finished its push motion, two features u and v, depicted as red circles, are randomly selected. From their trajectories S_u and S_v, a rigid transformation $\mathbf{A}_t$ 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 $A = \{\mathbf{A}_t\}_{t=0}^{T-1}$, 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.

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 [3] 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. The feature locations correspond to responses of the Shi-Tomasi feature detector described in Sec. IV-A. When the robot's end effector interacts with the object, a Lucas-Kanade tracker [13] is used to compute the optical flow of the sparse 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.

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 [14], [15]. 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 approximation 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. 4), we randomly draw 2 tracked features u,v and estimate a sequence of rigid transformations A_1 from their optical flow motions as first model hypothesis. The feature trajectories S_i that can be explained well by A_1 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 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.

We bias the sampling of the 2 points (line 6) such that drawn feature pairs are not likely to be further apart than the typical object size. 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 typical object size in pixels.

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

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

Algorithm 1: Randomized feature trajectory clustering

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: $c^*_{m,i} := \text{argmin}_{\{1,..,k_m,..,K_m-1\}} r_{k_m,i}$, 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: $m^* := \text{argmin}_m \sum_{k_m=1}^{K_m} \frac{\sum_i r_{k_m,i} \cdot \mathbf{1}_{\left[c^*_{m,i}=k_m\right]}}{\sum_i \mathbf{1}_{\left[c^*_{m,i}=k_m\right]}}$
17 Return: $K := K_{m^*}$, $\mathbf{c} := \left[c^*_{m^*,i}\right]_{i=1}^{N}$

We use the sum of squared residuals over all time increments, $r_{k,i} = \sum_{t=0}^{T-1} \|\mathbf{p}_{i,t+1} - \mathbf{A}_{k,t}\mathbf{p}_{i,t}\|_2^2$, 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.

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, as done in quality-threshold clustering. For a small maximal number of objects we have shown that K, M can be small such that the computational complexity of our algorithm is much lower. We leave out the analysis in the paper for the sake of brevity.

VI. Experiments

A. Experimental Setup

Our system was deployed on Willow Garage's PR2 robot. Depth images were taken from a Kinect sensor mounted on the robot's head and the PR2's built-in 5-megapixel camera was used for capturing images for feature extraction and tracking (See Fig. 1).

B. Segmentation of Objects in Cluttered Scenes

We evaluated our system on eight tabletop scenes with the cluttered background shown in Fig. 5. For each scene, the original setup of objects, the detected contact point candidates and push directions, and the feature clusters after the first push cycle are shown in the respective row. The scene images are depicted in color even though the tracking uses features extracted from gray scale images.

For each scene we evaluated the effect on the segmentation outcome that pushing corner-based contact points and push directions had as compared to pushing randomly into objects in the scene. Random pushing was implemented in the way that we first compute the contour of the object group and the minimum enclosing circle using functions provided by OpenCV library. We choose the point to push by randomly sampling the points on the contour whereas the direction of the push is estimated towards the center of the enclosing circle. In the corner-based pushing experiments we also evaluated the correctness of detected contact points and push directions. After capturing the initial scene, the PR2 moved its gripper along the push direction until one or more objects were successfully segmented or we reached a maximum 10 number of pushes. After each $1cm$ of arm travel a new picture was taken for optical flow tracking of the feature set detected in the first frame.

For each of the scenes we carried out three segmentation runs for both corner-based pushing and random pushing and present the averaged results for three runs in Fig. 6. In every run the success of the segmentation was inspected visually by the human operator and the object was removed from the

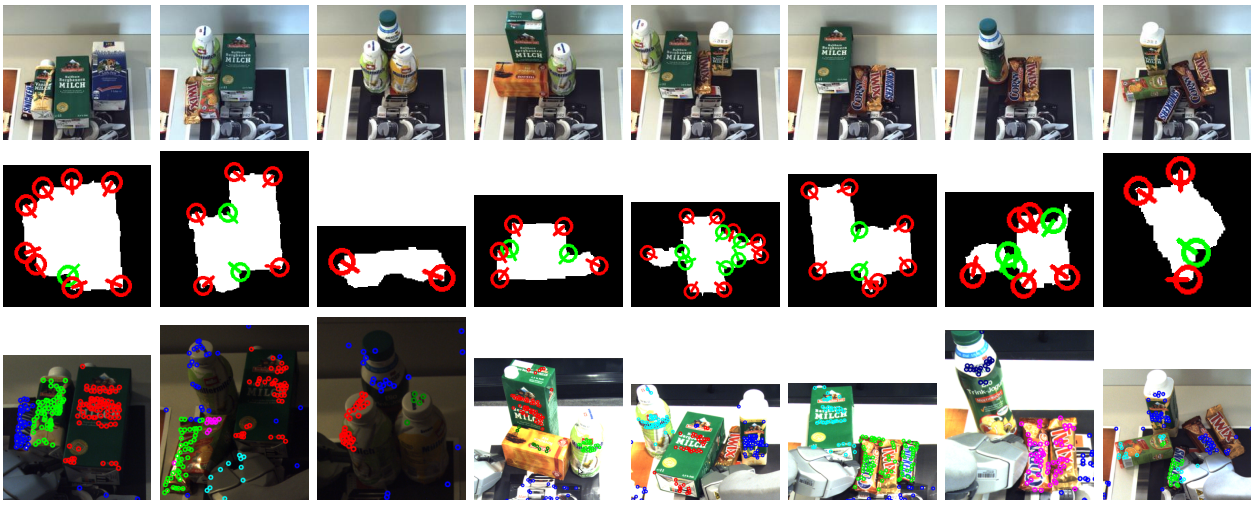

Fig. 5. Test scenes 1 to 8 from left to right. Top row: original scenes, middle row: contact point estimation, bottom row: segmentation after the first push cycle. Please note, that successfully segmented objects were removed from the scene and the contact point estimation and segmentation cycle were repeatedly executed.

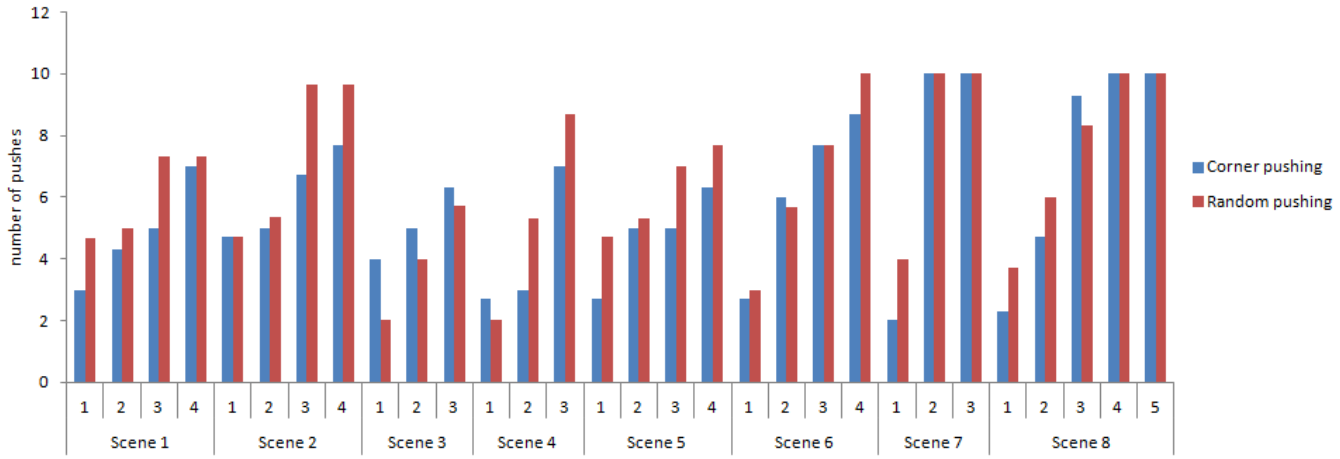

Fig. 6. Results on the segmentation of objects depicted in Fig. 5 using random vs. corner-based pushing. The tabular (upper) part of the figure denotes the average number of pushes over 3 runs needed to segment the respective object in the respective scene. Number 10 (maximum number of pushes allowed) 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.

scene by the operator upon the successful segmentation [2]. 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. 6 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 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 bottom of Fig. 7 where the push motion was induced such that all objects moved rigidly with respect to each other.

The segmentation of the scenes took 1.047 seconds on average to compute, which also demonstrates that our algorithm is suitable for real world settings.

C. Grasping

We also ran a grasping experiment on the scene 8 (Fig.5). In this experiment, we use low-quality image from the Kinect for the segmentation and 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 centroid to obtain the object position and then use the Principle Component Analysis to compute the orientation. The result of this experiment is presented

[2]Note that the scenes included several objects impossible to be grasped with only one PR2's arm or without a motion planner. We thus present grasping of objects from scene 8 in a proof-of-concept manner only (see Sec. VI-C)

a video accompanying this submission. For high resolution please refer to: http://youtu.be/4VVov6E3iiM.

D. Discussion

Fig. 7. Failure cases exemplified. In the left-top scene the "black pepper" object became occluded by the robot arm. In the right-top scene the features on the semi-transparent object were tracked unsuccessfully. Bottom row: failed segmentation from the random pushing experiment where the robot pushed objects such that they all moved rigidly with respect to each other

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 right 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.

VII. Textureless Object Segmentation

As pointed throughout this paper, the presented approach relies on the assumption of textured objects with Lambertian surfaces. In this section we report about our latest work with an aim to segment textureless objects using the very same randomized feature trajectory clustering algorithm listed in 1. While the current state of this work comprises of an alternative feature selection and tracking only, our preliminary results show promise for the robust and repeatable clustering as well (See Fig. 8, bottom). In a nutshell, we employ 3D line point cloud and 3D corner point cloud features (See Fig. 8, top) which we track using a particle filter. Both parts are compactly described below.

A. Extraction of 3D Line and Corner Point Clouds

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 Kinect sensor. Next we fit a line model to the object edge candidates using RANSAC algorithm [11] and finally pad the line with neighboring points on the object within a certain radius ($5cm$). 3D corner point clouds are determined using the 3D variant of the Harris corner detector as implemented in Point Cloud Library[3] and padded with neighboring points on the object within a certain radius ($5cm$) as well. Padding for both features is necessary in order to guarantee computation of a better likelihood function needed by the tracker as explained in the next section.

B. Particle Filtering-based Tracking

The feature point clouds extracted above are then passed to the particle filter-based tracker as reference models. The tracker itself consists of four steps: i) 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 an efficient tracking with a lesser number of the particles. In the re-sampling phase we utilize Walkers Alias Method [16]. The likelihood function of the hypotheses in the third step is computed based on the similarity between the nearest points pair of the reference point cloud and the input data. Similarity is defined as a product between a term depending on the points pair's euclidean distance and a term depending on points pair's match in the HSV color space. To obtain the model's weight we finally sum over likelihood values for every points pair in the reference model. Such likelihood function assures a combined matching of model's structure and visual appearance. In the final, normalization step we normalize the previously computed model weight by applying a relative normalization as described in [17]. 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 sampling [18] to select the optimal number of particles.

C. Results

We evaluated the robustness of the extraction and tracking of the 3D line point cloud and 3D corner point cloud features on the sequence of two textureless objects shown in the top part of Fig. 8 and video[4]. In bottom part of Fig. 8 we plot the relative distances over time between the features on one object and features on both objects respectively. The distance functions plotted in red and blue clearly show that the distance between the features on one object remains approximately the same while it changes significantly for the

[3] pointclouds.org
[4] http://youtu.be/oasorD8ZssE

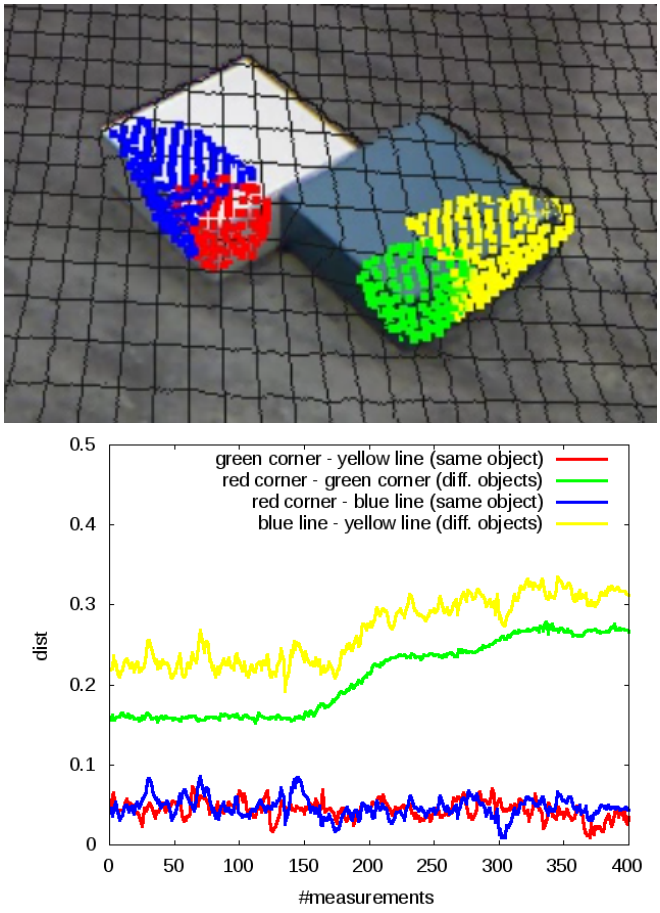

Fig. 8. Top: Two textureless objects and extracted features. Bottom: Evaluation of the change of relative distances (in m) between the features on above two objects during tracking. The incline of distance function at the measurement 170 reflects the movement of the white object while the incline at measurement 280 the movement of the blue object.

features on the different objects as depicted by the green and the yellow plot which is exactly suitable for our clustering algorithm. While these initial results are promising, we are aware that there is a myriad of issues we will have to tackle to achieve the desired robustness: from making the tracker more accurate, inferring the actual dense model of the object needed for e.g. grasping, to integrating it with the 2D feature-based tracker discussed in Sec. V, etc.

VIII. 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 and steady 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 scenes and successfully segmented them in more than 89% of cases. The results show applicability of our system for objects of various sizes, shapes and surface. We also show early results of our work towards the segmentation of textureless object using particle filter-based 3D point cloud feature tracking. In the future we plan to improve the latter and to integrate an arm motion and grasp planner [19] which will enable the robot to perform robust grasping and deal with more complex scenes.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

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

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

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

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

[19] 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.