POLITECHNIKA WARSZAWSKA
Wydział Mechatroniki

Praca dyplomowa magisterska

Karol Hausman

Interactive Segmentation of Textured and Textureless Objects in the Cluttered Scenes

Opiekun pracy:
Prof. dr hab. inż. Andrzej Masłowski

Jednostka dyplomująca:
Instytut Automatyki i Robotyki

Warszawa, 2012

Życiorys

Urodziłem się 21 czerwca 1988 roku w Koszalinie. W 2007 roku ukończyłem I Liceum Ogólnokształcące w Koszalinie. Bardzo dobre wyniki osiągnięte na egzaminie maturalnym pozwoliły mi podjąć naukę na Wydziale Mechatroniki Politechniki Warszawskiej. W 2011 roku otrzymałem tytuł inżyniera na specjalności Robotyka osiągając najlepsze wyniki w nauce na specjalności. Następnie podjąłem równocześnie studia magisterskie na Wydziale Elektroniki i Technik Informacyjnych na Politechnice Warszawskiej, studia licencjackie na Wydziale Filozofii i Socjologii Uniwersytetu Warszawskiego oraz studia magisterskie na Wydziale Mechatroniki. Po osiągnięciu bardzo dobrych wyników w nauce na wszystkich z powyższych kierunków postanowiłem wyjechać na studia zagraniczne. W październiku 2011otrzymałem prestiżowe stypendium rządu niemieckiego DAAD, które umożliwiło mi podjęcie studiów magisterskich Robotics, Cognition and Intelligence na Politechnice w Monachium, gdzie aktualnie studiuję.

Moją karierę zawodową zacząłem w roku 2010 od praktyk w firmie ASTOR, SIEMENS oraz MILKOMATIC, gdzie byłem projektantem systemów automatyki. Następnie kontynuowałem pracę w firmie MILKOMATIC do lutego 2011 na tym samym stanowisku. W marcu 2011 podjąłem pracę w firmie INNOTION, gdzie byłem programistą aplikacji na urządzenia mobilne. Po wyjeździe do Monachium kontynuuję swoją karierę zawodową w grupie badawczej Politechniki Monachijskiej o nazwie Intelligent Autonomous Systems Group, gdzie zajmuję się percepcją robotów humanoidalnych.

Karol Hausman

Instytut Automatyki i Robotyki
Politechnika Warszawska
ul. Św. Andrzeja Boboli 8
02-525 Warszawa

Institute of Automatic Control and Robotics
Warsaw University of Technology
Warsaw, Poland

Tel.: (+22) 849-96-16 Fax: (+22) 849-03-98 E-mail: iair@mchtr.pw.edu.pl URL: http://iair.mchtr.pw.edu.pl

Prowadzący pracę: prof. dr hab. inż. **Andrzej Masłowski**

Karol Hausman

Praca dyplomowa

,,Interactive Segmentation of textured and textureless objects in the cluttered scenes"

Założenia pracy:

a. Praca pisana w języku angielskim.
b. Podjęcie tematu dotyczącego aktualnego problemu percepcji robotów humanoidalnych.
c. Wykorzystanie sensora Microsoft Kinect.
d. Zbudowanie systemu zdolnego rozwiązać problem segmentacji w trudnych środowiskach.
e. Wykorzystanie humanoidalnego robota w celu weryfikacji systemu.
f. Wprowadzenie manipulacji w zakres działań robota.

Zakres pracy:

1. Przygotowanie systemu umożliwiającego interaktywną segmentację przedmiotów.
2. Implementacja wszystkich czynności poprzedzających właściwą segmentację, tj. manipulacja robotem, rozdzielenie przedmiotów.
3. Rozszerzenie systemu o obiekty nieposiadające znaków wymaganych do zastosowania technik prostego przetwarzania obrazu.
4. Testy systemu na humanoidalnym robocie

Abstract

This work 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. The approach on challenging scenes which included occluded objects, as well as objects of varying shapes and sizes was evaluated.

In the second part of this work new approach towards segmentation of textureless objects was presented. The proposed system allows a robot to effectively segment textureless objects in cluttered scenes also by leveraging its manipulation capabilities. In this threefold approach, the cluttered scenes are first statically segmented using part-graph-based hashing and then the interactive perception is deployed in order to resolve possibly ambiguous static segmentation. In the second step the RGBD (RGB + Depth) features, estimated on the RGBD point cloud from the Kinect sensor, are extracted and tracked while the motion is induced into a scene. The algorithm also autonomously infers appropriate movements which can effectively separate objects. In the final step, the resulting tracked feature trajectories are assigned to their corresponding object by using a graph-based clustering algorithm, which edges measure the distance dissimilarity between the tracked RGBD features. The approach was evaluated using the PR2 robot on a set of scenes which consist of textureless flat (e.g. box-like) and round (e.g. cylinder-like) objects.

Streszczenie

Tematem pracy jest opis systemu umożliwiającego segmentację przedmiotów w nieuporządkowanych środowiskach. System opiera się na wprowadzeniu manipulacji do zakresu działań robota. Pierwsza część pracy opisuje algorytm dotyczący przedmiotów o zwartej strukturze. Algorytm ten składa się z kilku etapów. Robot początkowo stara się znaleźć wklęsłości na płaszczyźnie posadowienia przedmiotów. Jako że przedmioty są przeważnie wypukłe, jest bardzo prawdopodobne, że robot kierując chwytak we wklęsłe miejsce rozdzieli dwa obiekty. Następnie znajdowane są punkty w opisanym środowisku, które są na tyle stabilne, aby można je było śledzić w trakcie ruchu przedmiotów. Robot rozdziela dwa przedmioty powodując ich ruch w przeciwnych kierunkach. W trakcie tego etapu wszystkie punkty są śledzone i grupowane w klastry, które definiowane są poprzez te same jednorodne przekształcenia. Klastry są efektem końcowej segmentacji obiektów.

Druga część pracy poświecona jest obiektom, na których nie można znaleźć punktów, które są łatwe do śledzenia. W tym wypadku początkowe etapu systemu są podobne, jednak różnice są w cechach obrazu, które są obserwowane. Wyodrębniane są trójwymiarowe cechy chmury punktów takie jak krawędzie czy narożniki, które następnie są obserwowane z wykorzystaniem algorytmu opartego na filtrze cząsteczkowym. Algorytm umożliwiający grupowanie cech jest również różny od procedury opisanej w pierwszej części pracy. Jest on bowiem oparty na grafie, gdzie cecha jest reprezentowana jako wierzchołek grafu, a odległość do innych cech obrazuje krawędź grafu. W ten sposób, wystarczy tylko jedna cecha na jednym obiekcie, aby móc go prawidłowo rozróżnić.

Obydwa systemy zostały przetestowane podczas eksperymentów z różnymi przedmiotami na robocie humanoidalnym PR2, który został zbudowany przez firmę Willow Garage.

Table of Content:

I. INTRODUCTION 7

II. RELATED WORK 9

III. SYSTEM ARCHITECTURE 11

IV. ESTIMATION OF CONTACT POINT AND PUSH DIRECTION 12

A. Contact Points from Concave Corners. 12

B. Push Direction and Execution 13

V. TEXTURED OBJECT SEGMENTATION USING FEATURE TRAJECTORIES 15

A. Feature Trajectory Generation using Optical Flow 15

B. Randomized Feature Trajectory Clustering with Rigid Motion Hypotheses 15

C. Trajectory Clustering Complexity Analysis 19

VI. EXPERIMENTS 20

A. Experimental Setup 20

B. Segmentation of Objects in Cluttered Scenes 20

C. Grasping 22

D. Discussion 23

VII. TEXTURELESS OBJECT SEGMENTATION 24

VIII. MOTIVATION FOR TEXTURELESS OBJECTS 25

IX. STATIC OBJECT PRE-SEGMENTATION USING ADDITIVE FEATURES AND HASHING OF PART-GRAPH DESCRIPTORS 29

A. Decomposition into Part Graphs 29

B. Object Part Categorization 30

C. Verification of Correctness of Segmentation 30

X. TEXTURELESS OBJECT SEGMENTATION 32

A. RGBD Features 32

B. Particle Filtering-based Tracking of RGBD Point clouds 33

C. Trajectory Clustering 35

XI. EVALUATION AND DISCUSSION 36

A. Experimental Setup 36

B. System Pipeline 37

C. Results 38

D. Discussion 39

XII. CONCLUSIONS AND FUTURE WORK 40

References: 41

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 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], a system that uses a robot arm to induce motions in a scene to enable effective object segmentation is proposed. This system employs a combination of the following techniques:

1) estimation of a contact point and a push direction of the robot's end effector by detecting the concave corners in the cluttered scene,
2) feature extraction using features proposed by Shi and Tomasi [3] and tracking using optical flow [4], and
3) 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, it was shown that the 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. The system is available as opensource and can be deployed on a robot equipped with a 2D-camera, a depth camera and at least one arm. The first part of this thesis provides the following main contributions:

- 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 second part presents the solution for objects without the texture. The deeper look into this problem can be found in section VII.

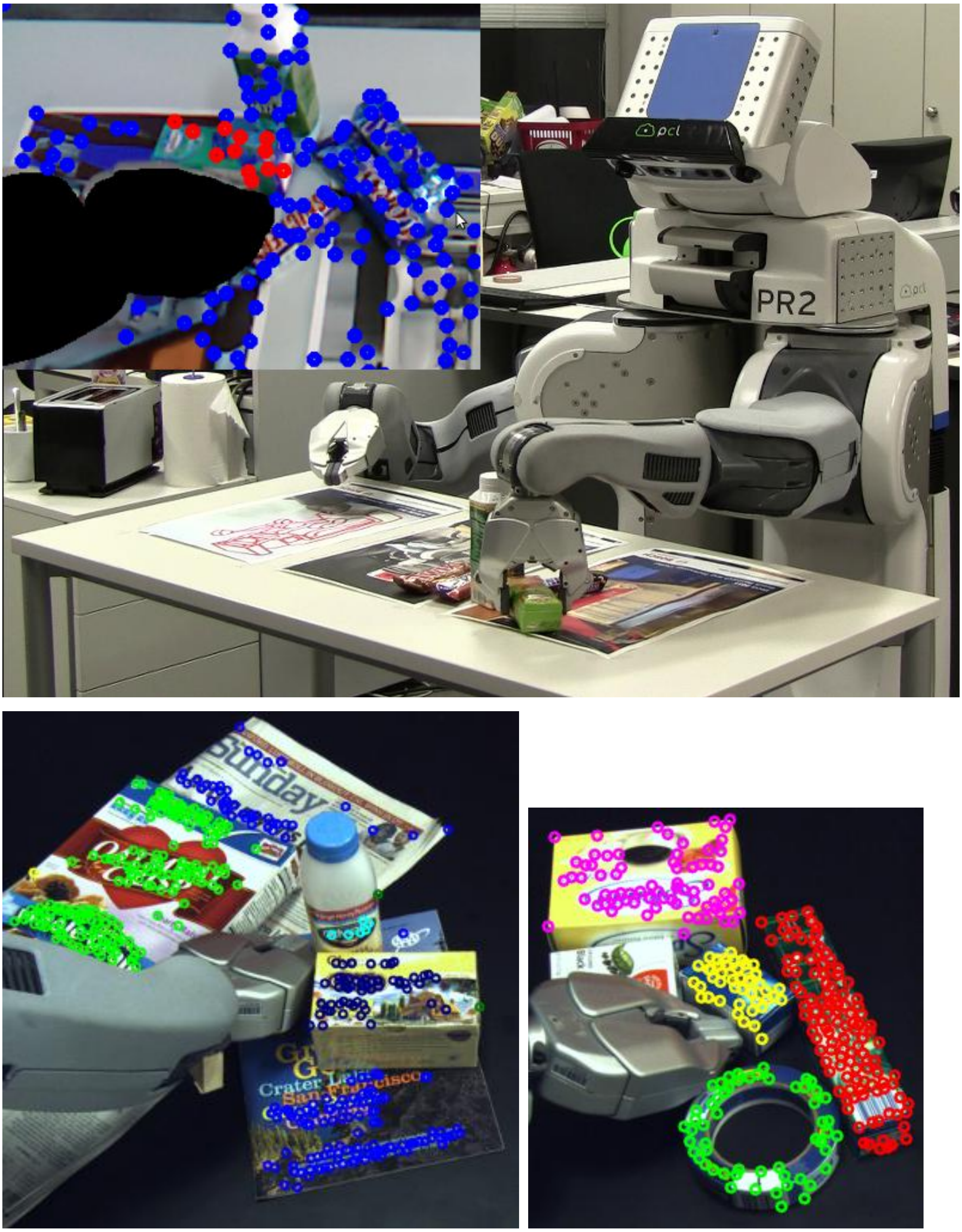


Fig. 1. Top: PR2 robot successfully picking-up the object after segmenting in it in clutter using herein proposed object segmentation algorithm. Bottom: the result of clustering of two highly cluttered scenes.

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 work 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 [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, the arm motion is not pre-planned but adapts to the scene, the 3D data was used to segment the object candidates from the background and a novel clustering approach for the segmentation of textured objects is proposed. 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 cannot deal with heavily textured objects but could work well in combination with this work.

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 this thesis 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 is detected, the robot continues moving its end effector a predefined distance along the push direction. Since it can be assumed that the robot arm is calibrated, a self-filter was employed 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.

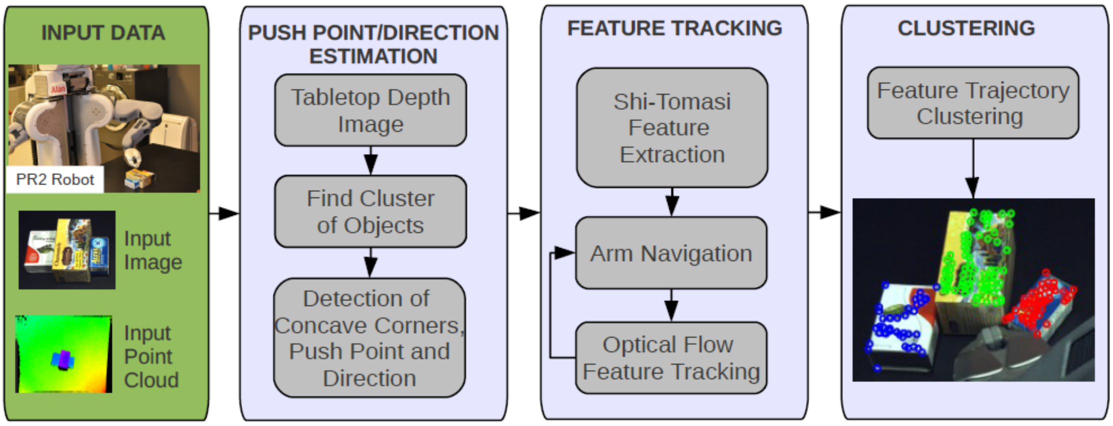

INPUT DATA
PR2 Robot
Input Image
Input Point Cloud
PUSH POINT/DIRECTION ESTIMATION
Tabletop Depth Image
Find Cluster of Objects
Detection of Concave Corners, Push Point and Direction
FEATURE TRACKING
Shi-Tomasi Feature Extraction
Arm Navigation
Optical Flow Feature Tracking
CLUSTERING
Feature Trajectory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

Fig. 2. The system proposed in the thesis 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.

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, a cluttered tabletop scene is considered. Since most commonly encountered household items have convex outlines when observed from above, the 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, a heuristic to determine a contact point and a push direction from depth-sensor data is described.

A. Contact Points from Concave Corners.

The problem of finding a contact point is restricted to the table plane. The 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, standard morphological operators and 2D-contour search [12] were employed 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], the corner response for each pixel location $p = (p_x, p_y)$ was computed, based on the covariance matrix Z:

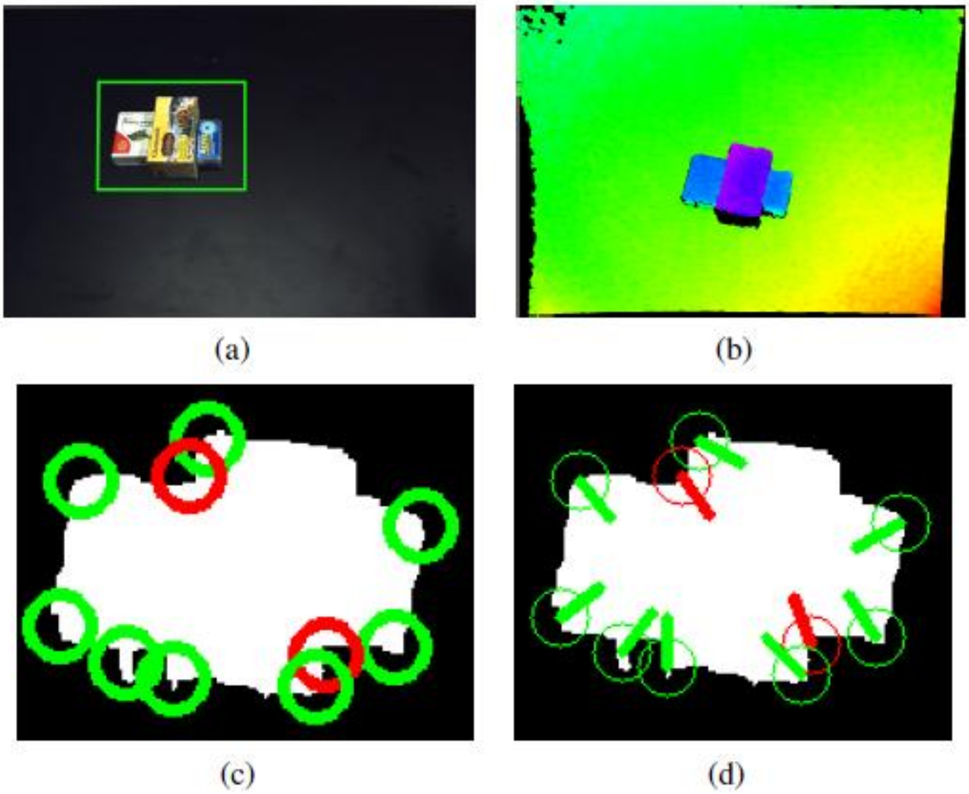
$$\mathbf{Z}(\mathbf{p}) = \begin{bmatrix} \sum\limits_{S(\mathbf{p})} I_x^2 & \sum\limits_{S(\mathbf{p})} I_x I_y \\ \sum\limits_{S(\mathbf{p})} I_x I_y & \sum\limits_{S(\mathbf{p})} I_y^2 \end{bmatrix} ,$$

where I_x and I_y are the image gradients in x and y direction and S(p) the neighborhood of p.

A corner detector response is recorded if $min(\lambda_1, \lambda_2) > \theta$ where λ_1, λ_2 are the eigenvalues of Z and θ is a given threshold. There is also a check for roughly equal eigenvalues λ_1, λ_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 Z(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).


(a) (b)

(c) (d)

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.

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 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

One can take advantage of the objects' texture properties by extracting $i = 1 \ldots . N$ 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 $p_{i,t}$ 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, the usage of the rigid body property of objects can be taken and it can be assumed that each subset of the features trajectories 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. features are clustered with respect to how well rigid transformations can explain their motions. As the objects only move on the table plane, a possible rigid transformation A is restricted to be composed of a 2D-rotation R, a 2D-translation t and a scaling component, i.e. s $\mathbf{A} = s \cdot [\mathbf{R} \mid \mathbf{t}]$.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.

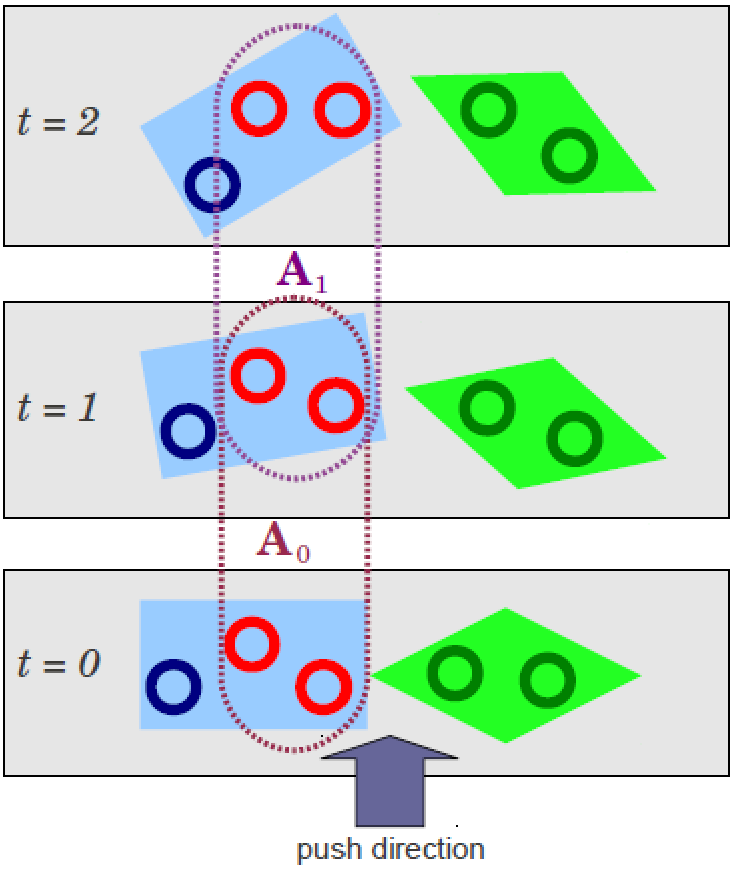


Fig. 4. Feature trajectory clustering with rigid motion hypotheses: Each feature i, depicted as a circle, is tracked over each time step t, forming a trajectory of feature positions S_i. After the robot finished its push motion, two features u and v, depicted as red circles, are randomly selected. From their trajectories S_u and S_v, a rigid transformation 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 proposed clustering algorithm 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), two tracked features u, v are drawn and from their optical flow motions a sequence of rigid transformations A_1 can be estimated 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 two 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. The sampling of the 2 points was biased (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 p_i and p_u:

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

where σ is set to half of the typical object size in pixels.

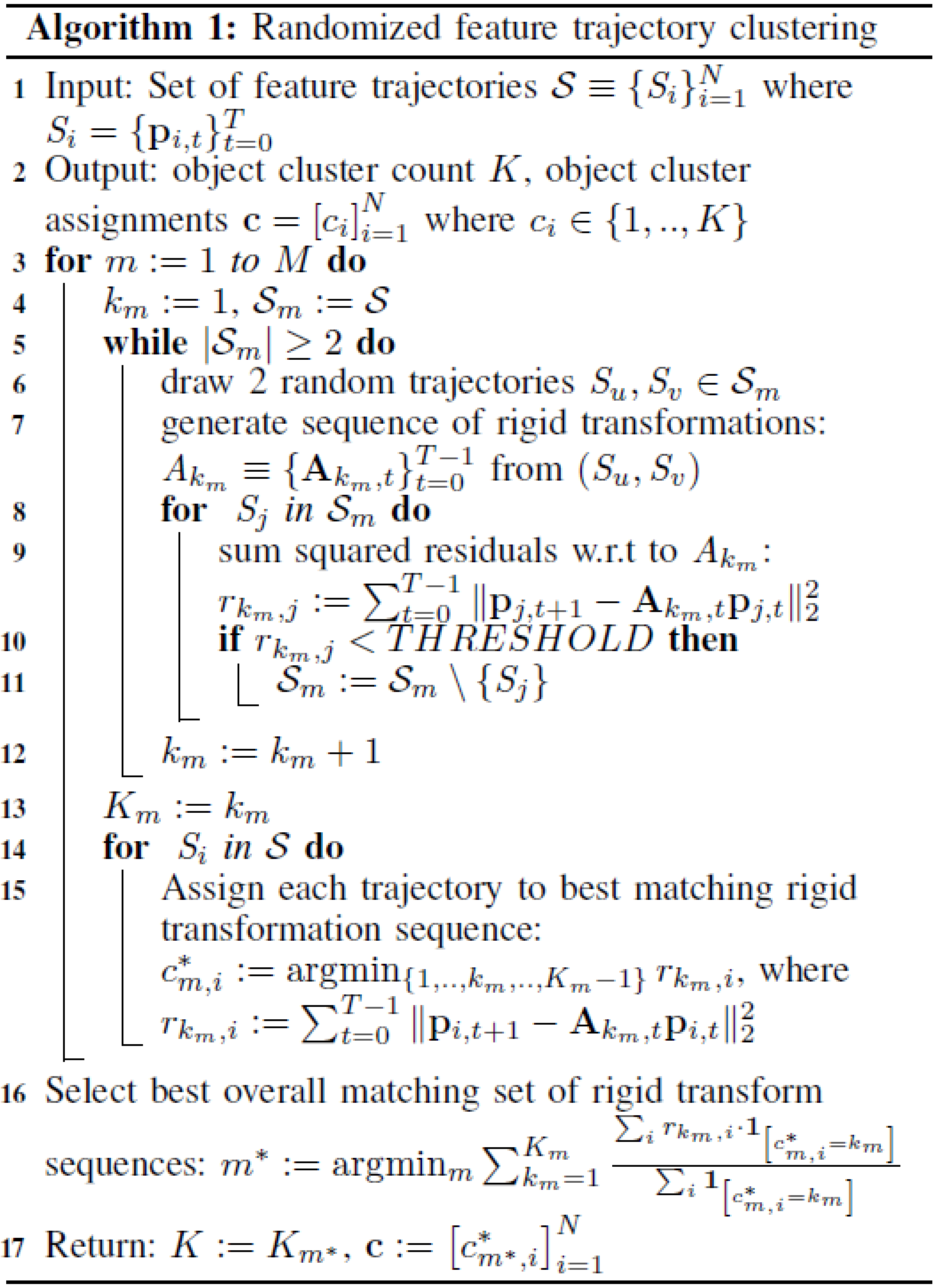
In line 7, a rigid transformation A_t is computed from the trajectories S_u and S_v at each time increment from t to t + 1. A 4-DOF transformation At can be computed directly using geometrical considerations from the two 2D-feature point locations p_u and p_v at t and t + 1, such that:

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

The sum of squared residuals over all time increments was used,

$$r_{k,i} = \sum_{t=0}^{T-1} \| \mathbf{p}_{i,t+1} - \mathbf{A}_{k,t} \mathbf{p}_{i,t} \|_2^2$$

, as a measure of how well a feature trajectory Si 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 $\mathbf{A}_{k,t}$.



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

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. The best m is chosen 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 it can be shown that K, M can be small such that the computational complexity of the algorithm is much lower.

VI. EXPERIMENTS

A. Experimental Setup

The 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

The system was evaluated 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 the effect on the segmentation outcome was evaluated 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 one had to first compute the contour of the object group and the minimum enclosing circle using functions provided by OpenCV library.

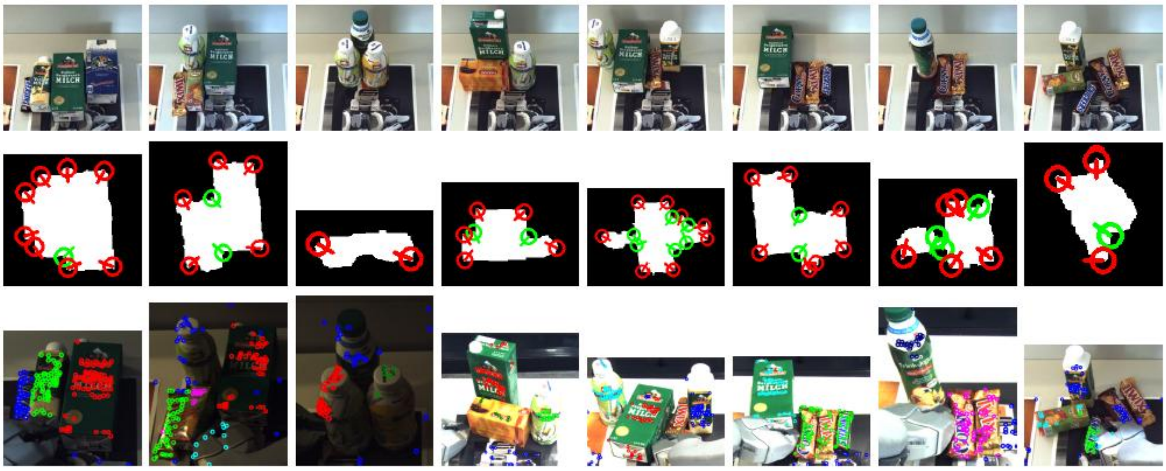


Fig. 5. Test scenes 1 to 8 from left to right. Top row: original scenes, middle row: contact point estimation, bottom row: segmentation after the first push cycle. Please note, that successfully segmented objects were removed from the scene and the contact point estimation and segmentation cycle were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

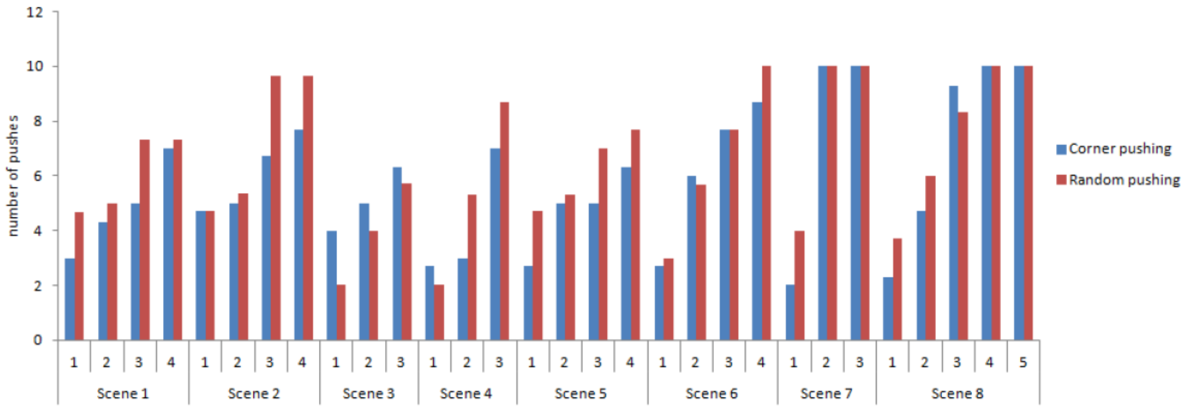


Fig. 6. Results on the segmentation of objects depicted in Fig. 5 using random vs. corner-based pushing. The tabular (upper) part of the figure denotes the average number of pushes over 3 runs needed to segment the respective object in the respective scene. Number 10 (maximum number of pushes allowed)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

The point to push was chosen 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 the correctness of detected contact points and push directions was also evaluated.

After capturing the initial scene, the PR2 moved its gripper along the push direction until one or more objects were successfully segmented or a maximum 10 number of pushes was reached. 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 out three segmentation runs were provided 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 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. 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 there were only 3 unsegmented scenes and total of 10 nonsegmented 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 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 the algorithm is suitable for real world settings.

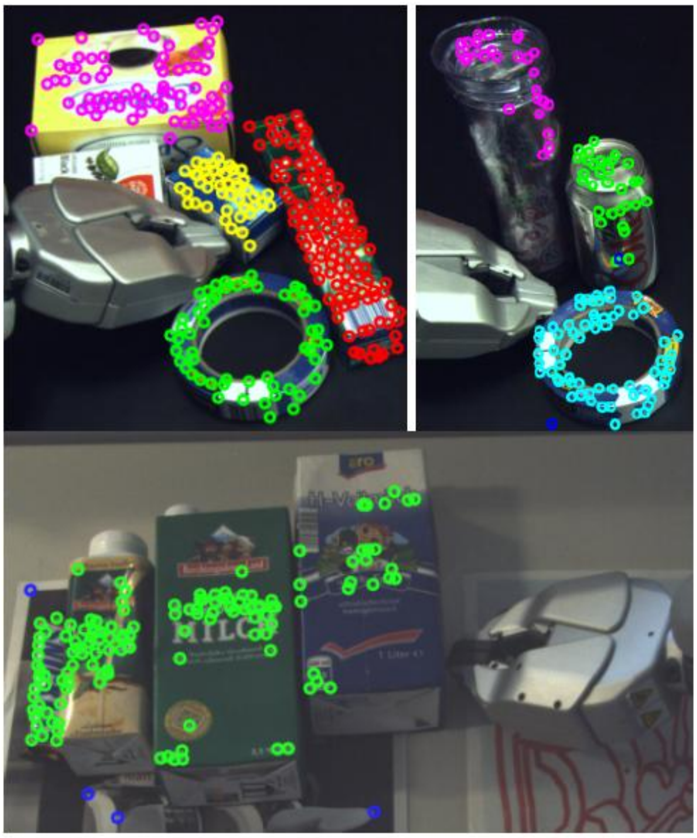


Fig. 7. Failure cases exemplified. In the left-top scene the "black pepper" object became occluded by the robot arm. In the right-top scene the features on the semi-transparent object were tracked unsuccessfully. Bottom row: failed segmentation from the random pushing experiment where the robot pushed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

C. Grasping

There was also provided a grasping experiment on the scene 8 (Fig.5). In this experiment, low-quality image from the Kinect was used for the segmentation and an associated point cloud for the calculation of the object pose. To compute the latter the set of 3D points corresponding to the the set of 2D features from the successfully segmented cluster was taken and an Euclidean clustering was applied to remove possible outliers. After that the centroid was computed to obtain the object position and then the Principle Component Analysis was used to compute the orientation. For high resolution video of the experiment please refer to: http://youtu.be/4VVov6E3iiM.

D. Discussion

To exemplify some of the failed cases it is crucial to direct 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 work, the presented approach relies on the assumption of textured objects with Lambertian surfaces. In this section the latest work is reported with an aim to segment textureless objects.

The proposed system allows a robot to effectively segment textureless objects in cluttered scenes by leveraging its manipulation capabilities. In this threefold approach, the cluttered scenes are first statically segmented using part-graph-based hashing and then the interactive perception is deployed in order to resolve possibly ambiguous static segmentation. In the second step the RGBD (RGB + Depth) features, estimated on the RGBD point cloud from the Kinect sensor, are extracted and tracked while the motion is induced into a scene. In the final step, the resulting tracked feature trajectories are assigned to their corresponding object by using a graph-based clustering algorithm, which edges measure the distance dissimilarity between the tracked RGBD features. The approach is evaluated again by using the PR2 robot on a set of scenes which consist of textureless flat (e.g. box-like) and round (e.g. cylinder-like) objects.

VIII. MOTIVATION FOR TEXTURELESS OBJECTS

Why is it important to take care of textureless scene? For example, when setting the table the robot is likely to be confronted with a cluttered unstructured scene like the example shown in Fig. 8. Why interactive perception? The need for this work is motivated by looking at the tabletop scene depicted in Fig. 8. In Fig. there are presented three state-of-the-art segmentation algorithms operating in depth, RGB and RGBD space respectively on the given scene. It is noticed that the results are far away from being optimal in the cases of:

a) same color objects (a coffee cup and a saucer),
b) similar shape objects and occlusions (a white and a blue box) and also in the case of
c) a sensor default (cutlery in this case appears transparent to the Kinect sensor).

While one can certainly fine tune the algorithms’ parameters for a certain setup and environment none of the approaches will generalize. 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. In this work there is a focus on proposing the solution for the cases a) and b) and limit to the flat and round objects.

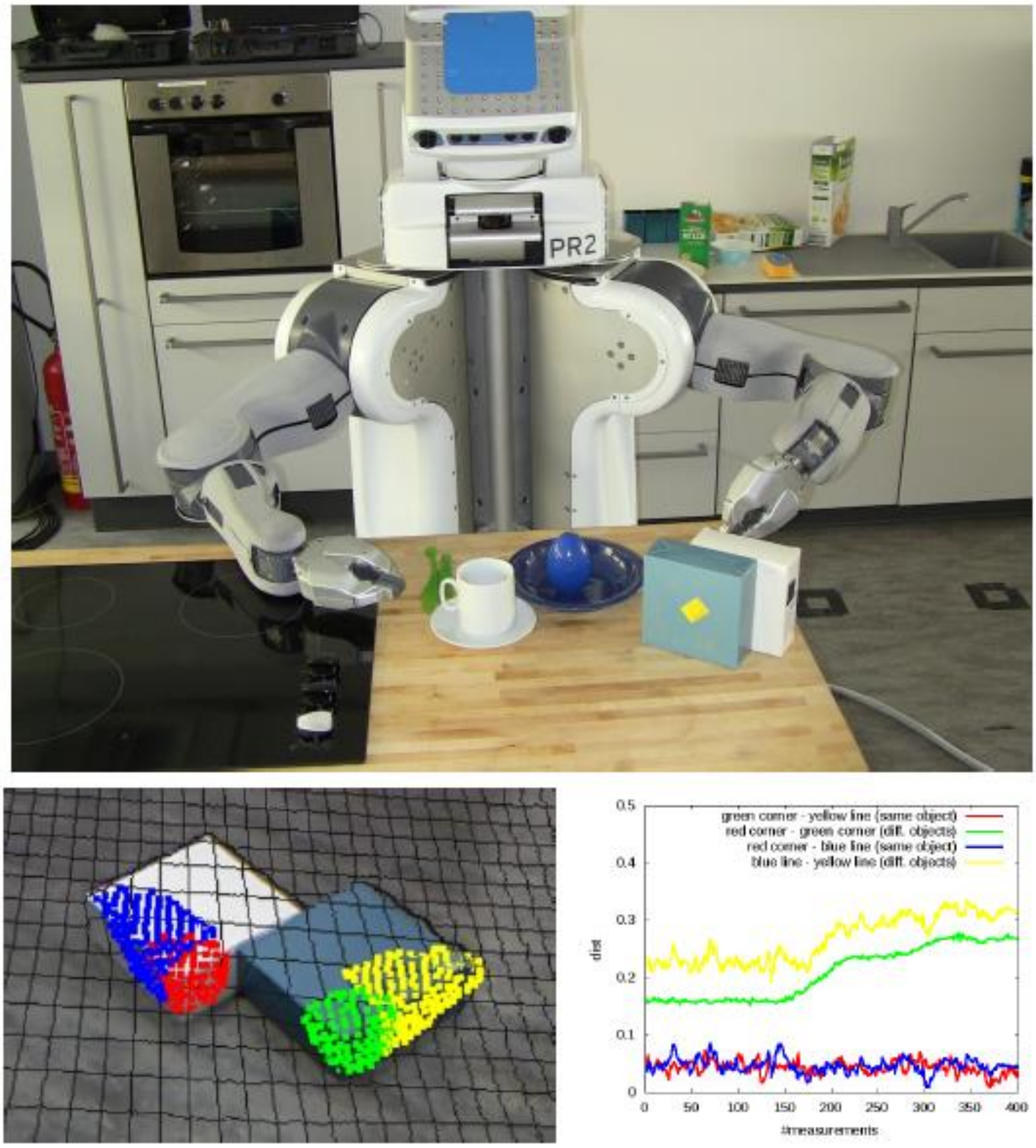


Fig. 8. Top: The service robot PR2 aiming to segment the scene consisting of textureless object. Bottom-left: Two flat textureless objects from the above scene and extracted features. Bottom-right: Evaluation of the change of relative distances (in m) between the features on above two objects during 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.

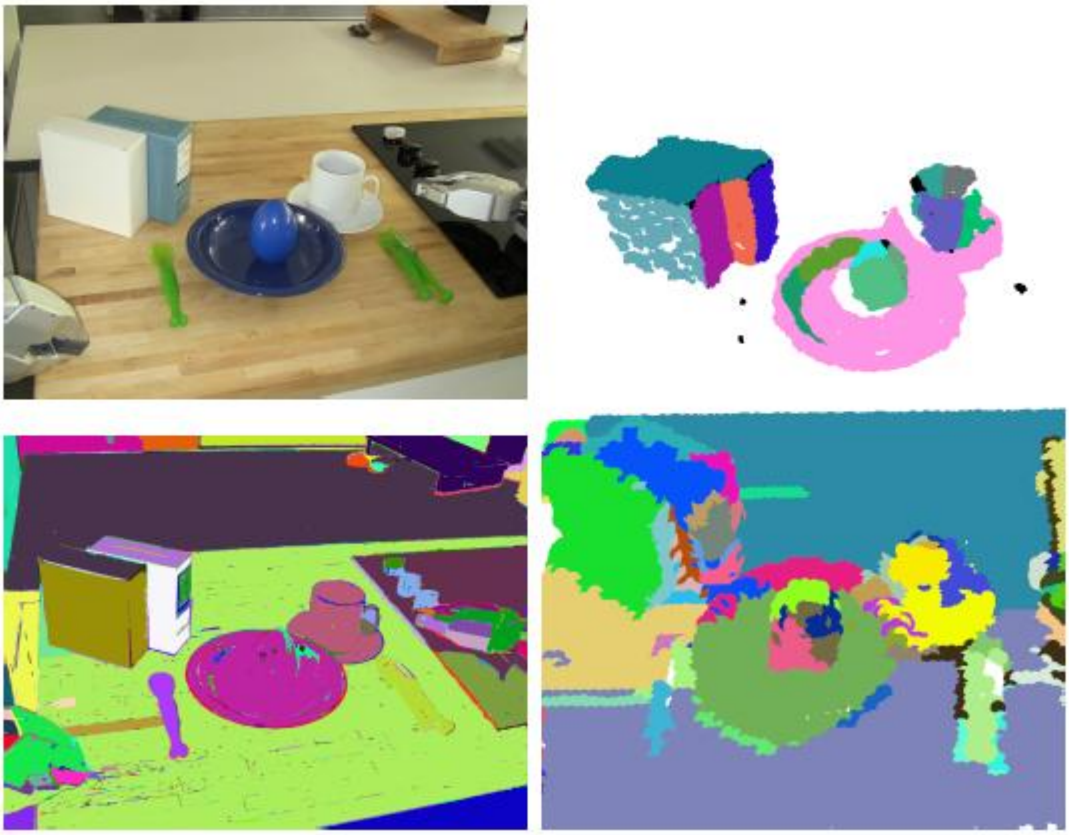


Fig. 9. Results of the scene (top-left) segmentation using Part-Graph-based Hashing [17] method (top-right), Graph-based method [18] (bottom-left) and the Region Growing method [19] (top-right). These methods work in depth, RGB and RGBD space respectively and all results exhibit the nature of the difficulty of this task.

A system proposed here uses a robot arm to induce motions in a scene to enable effective object segmentation. The system employs a combination of the following techniques:

i) pre-segmentation from the single camera view using part-graph-based hashing [17] and estimation of a contact point and a push direction of the robot's end effector by detecting the concave corners in the cluttered scene [16],

ii) feature extraction using RGBD features described in Sec. V-A and tracking using particle filtering-based tracking as presented in Sec. V-B and

iii) graph-based trajectory clustering algorithm elaborated in Sec. V-C. An important assumption in the system is in that each item is a rigid body and not subject to deformations when interacting with the robot's end effector or other objects.

Evaluation was performed on several scenes with flat and round objects of similar colors, shapes and sizes. In overall these sections provide the following novel contributions for the segmentation of scenes consisting of textureless tabletop objects:

- A set of carefully selected RGBD features suitable for the tracking of flat and round textureless object;
- A particle filtering-based tracking algorithm for above features;
- A graph-based algorithm for the clustering of 3D-feature trajectories which edges measure the dissimilarities between the RGBD features’ distances;
- An inclusion of the static pre-segmentation algorithm and integration of all above into a pipeline using ROS (Robot Operating System) as depicted in Fig. 13.

IX. STATIC OBJECT PRE-SEGMENTATION USING ADDITIVE FEATURES AND HASHING OF PART-GRAPH DESCRIPTORS

In order to achieve a pre-segmentation the classification method presented in [17] based on part-graph-based hashing is used. The basic idea of that work is that segmenting objects accurately in a cluttered scene does not always yield the expected result as seen in Fig. 11, row 4 and can lead to classification failures, but over-segmenting is easily realizable [20]. A method for classifying objects in cluttered scenes based on these object parts can be used by considering combinations of parts to compute features and classifying these efficiently aided by hashing.

Why using a categorization algorithm for pre-segmentation? In Fig. 9 it was alluded to the fact that it is close to impossible to recover the failure cases since the graph-based segmentation algorithm as well as the region growing one work on the level of pixels and points in the point cloud. Part-graph-based hashing algorithm on the other hand is a learning-based algorithm that bears the semantic of the object categories with it. In this particular case it is possible to collect the statistics about the probable number of parts one object is composed of in the training stage and then deduce whether the categorization of a given scene is likely correct or not. In the rest of the section the part-graph-based hashing algorithm is summed up briefly and it is shown how it is turned into a “rich” segmentation algorithm.

A. Decomposition into Part Graphs

In order to find the parts ($p_1,\ldots,p_n \in P$,P being a list of all parts) of the scans the clustering criteria presented in [20] is used, such that patches with a relatively small curvature are considered, as shown in Fig. 11 row 3. Small patches are discarded, and for each part a feature is subsequently computed and stored for later use. Then the part neighborhoods are extracted by checking if the physical distance between two parts falls below a threshold, and build a connectivity matrix. Starting at each vertex, all the possible groupings up to a certain size are created in order to obtain the “soup of segments”. Afterwards, the groups’ hash codes are formed by using isomorphic graph metrics. For a detailed description of this and the region growing used please refer to [17].

B. Object Part Categorization

The classifier was trained on a subset of the dataset from [21] as presented in [17]. The choice of the feature determined for each part, namely the GRSD- (Global Radius-based Surface Descriptor [22]) feature is motivated by the fact that there are the novel objects not seen before by the classifier, so in order to successfully categorize them one needs to use geometric features. As described in the previous work, objects ($o_1,\ldots,o_n \in O$, O being a list of all object hypotheses) are categorized in six geometrical categories: sphere, box, flat, cylinder, disk/plate and other. Because of doing this, there is a better discrimination between different objects. After the results for the six geometrical classes are collected, they are merged 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 a three class classification problem is obtained. Tested on a separate subset of the database from [21] a general classification rate of 90% in the case of the six geometric classes and 94.5% for the three class problem are obtained. In this work the category other is omitted and the other two are used in order to determine which RGBD feature 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 for voting for object centroids followed by a model fitting step was described in [20], but no CAD models for test objects in this thesis are assumed. Also 6-DOF poses have to be considered, complicating the approach considerably.

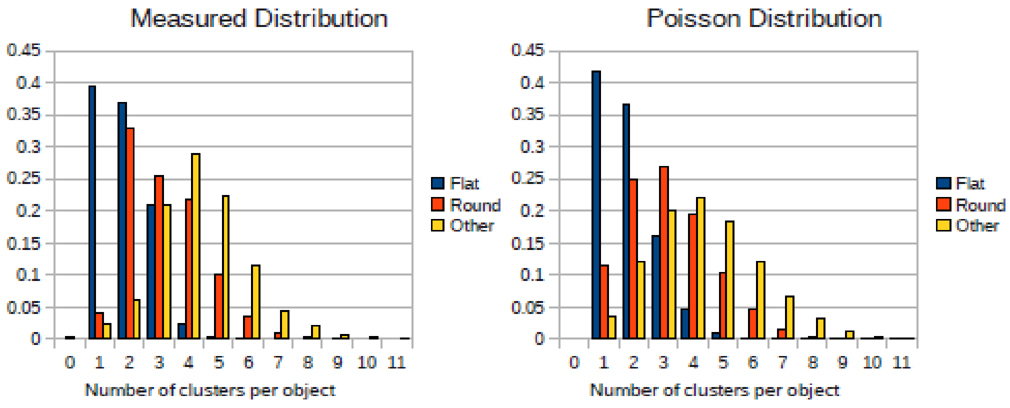


Fig. 10. Distribution of number of parts (Fig. 11 row 3) per object (for different object types) and their approximation with a Poisson distribution.

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. 10, the distribution of the number of different object parts, generated in the training stage of the above summed up part-graph-based hashing algorithm, can be modeled as a Poisson distribution, with an average error of 1.7% (and at most roughly 9%). The Poisson distribution described below describes the probability of different number of events occurring in a given interval, which are interpreted here as the number of part boundaries encountered over the surface of the scanned object. The parameter is the mean of number of parts, which in this case is 0.876 for flat, 2.166 for round, and 3.317 for other object types.

$$P(k\ parts\ forming\ a\,single\,object) = \lambda^{k} \exp - \lambda / k!$$

This simple model can be 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. Since the distributions for the geometric categories that were grouped into the three types was very similar, the evaluation of these probabilities gets simplified. Example: To demonstrate this, from the Fig. 10 it can be deduced that the flat object is most likely to consist of 1 or 2 parts. The test scene with 2 boxes (Fig. 11) was categorized as one object (row 4), but in row 3 it can be observed that this region consists of 6 parts which clearly indicates an over-segmentation error and the need for the robot to segment this region interactively.

X. TEXTURELESS OBJECT SEGMENTATION

In this section it is reported about the selected RGBD features suitable for the tracking of textureless objects and the particle filtering-based tracking library. The features are estimated on the categorized part of the RGBD point cloud from the Kinect sensor where the RGB and the depth measurements are time synchronized and registered. 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 are employed. The rationale behind this selection of features is that they are all fast to compute and yet distinctive enough for tracking with the below proposed tracking algorithm. The latter uses a combination of a visual appearance and a 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 one has to first find object edge candidates in the cluttered scene using curvature values computed in the input point cloud from Kinect sensor. Next one needs to fit a line model to the object edge candidates using RANSAC algorithm 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[4] 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 subsection. The features are shown in Fig. 11 row 2 and 5, 1st column. To obtain a 3D cylinder point cloud, also RANSAC algorithm is used, which is based on the observation that on a cylinder surface, all normals are orthogonal to the cylinder axis, and intersect it. One has to 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 enforcing an upright position results are more robust, but it is not mandatory. Finally, the generation of the 3D circle is also done using RANSAC and based on the idea that a sample point is projected into the 3D circle's plane and 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 is computed, whereas the line is passing through the projected sample point. The features are shown in Fig. 11 row 2 and 5, 2nd column.

B. Particle Filtering-based Tracking of RGBD Point clouds

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 one has to use a ratio between a constant position and a constant velocity motion model which allows to achieve an efficient tracking with a lesser number of the particles. In the re-sampling phase Walkers Alias Method [24] is utilized. 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 (Hue, Saturation, Value) color space. To obtain the model's weight one has to 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 the previously computed model weight is normalized by applying a relative normalization as described in [25]. 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 [26] to select the optimal number of particles. Although the tracking is done for 6 DOF in this thesis only the translations are used and it is planned to incorporate the orientations in the future. Why not to track object parts? To answer this question the reader has to be referred to the scene 3 in Fig. 11, row 4 where a cup was categorized as a flat object. Under the induced motion, the flat object hypothesis and thus the reference cloud in the tracking algorithm would break up and result in a lost and irrecoverable track.

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                               */
$A$ = getConenctions($M_{breaks} <= p_{threshold}$)
/* number of clusters based on Laplacian                */
$nr_{clusters}$ = nrZeros(eigenValues(diag(degrees($A$)) - $A$))
/* get features clustered by connectivity               */
Output: $F_{clusters}$ = connectedComponents($A$)
```

C. Trajectory Clustering

The tracked features' 3D trajectories (see Fig. 11 row 6) are clustered using Alg. 2. One has to 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). The final connection matrix is then obtained by removing those edges whose weight exceeds a given percentage ($p_{threshold}$) of the theoretic maximum (m). This way those features are connected whose distance did not vary too much for significantly long.

XI. EVALUATION AND DISCUSSION

A. Experimental Setup

The system was evaluated on 11 scenes, three of which are presented in greater detail in Fig. 11, consisting of combinations of up to 4 flat and round objects. Both the tracking and the clustering algorithm perform linearly with the number of features and objects respectively and can thus easily be used for larger and more complex scenes.

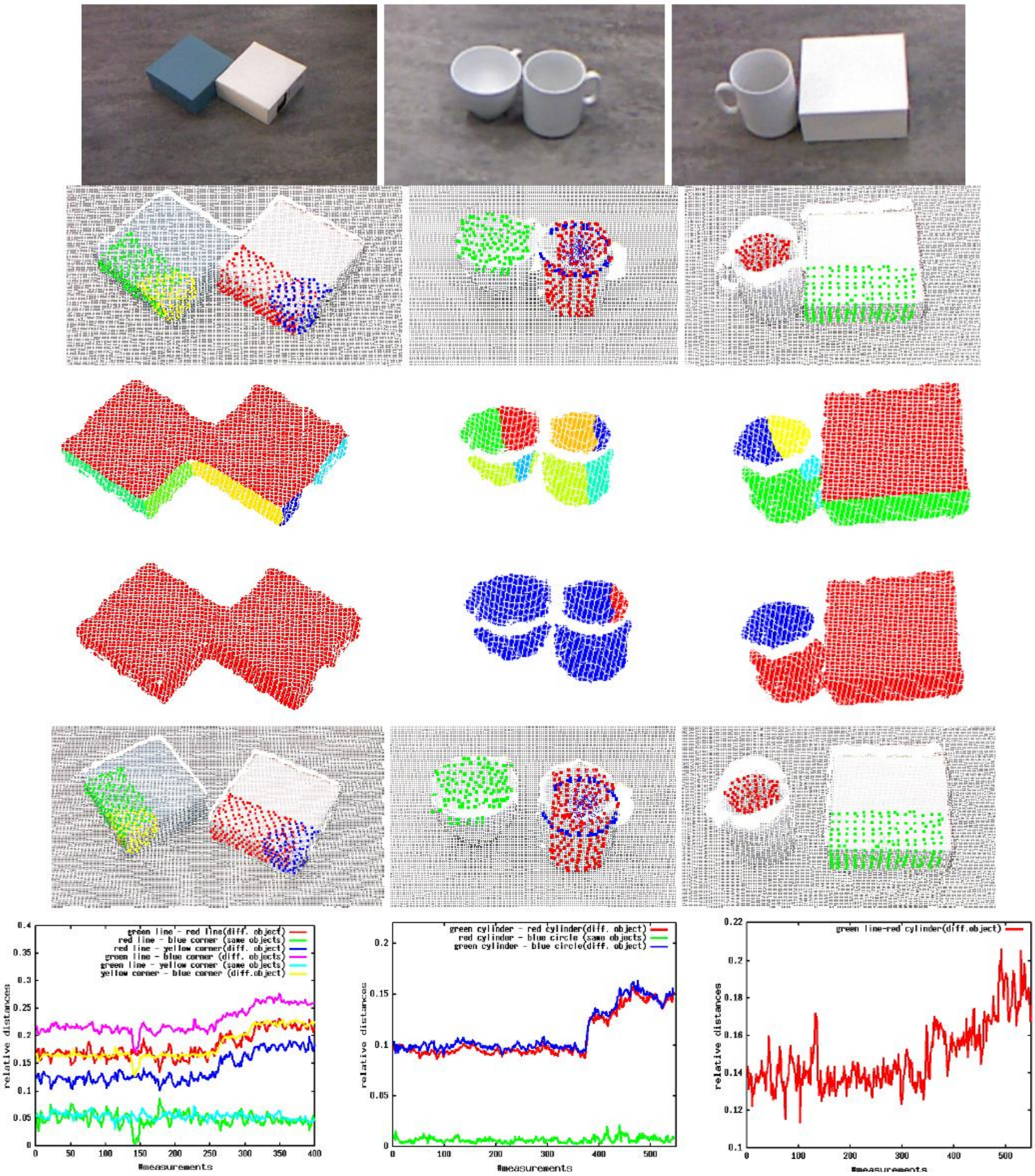


Fig. 11. Three test scenes 1 to 3 from left to right. First row: original scenes, second row: extracted RGBD features before the interaction, third row: parts P from the static segmentation, fourth row: object hypotheses O from the static segmentation, fifth row: tracked RGBD features after interaction, sixth row relative distances between the tracked features.

Occlusions, self-occlusions and stacked objects are planned to be investigated in the future research for which the current system is a great starting point.

B. System Pipeline

The approach presented in this thesis is realized as part of the Robot Operating System open source framework, and primarily consists of three main nodes as depicted in Fig. 12. In the first node a point cloud from the Kinect sensor is obtained and static object pre-segmentation is performed. As a result a set of categorized object hypotheses O is obtained, with the category being either flat or round, and a list of object parts P_o that every object $o \in O$ consists of.

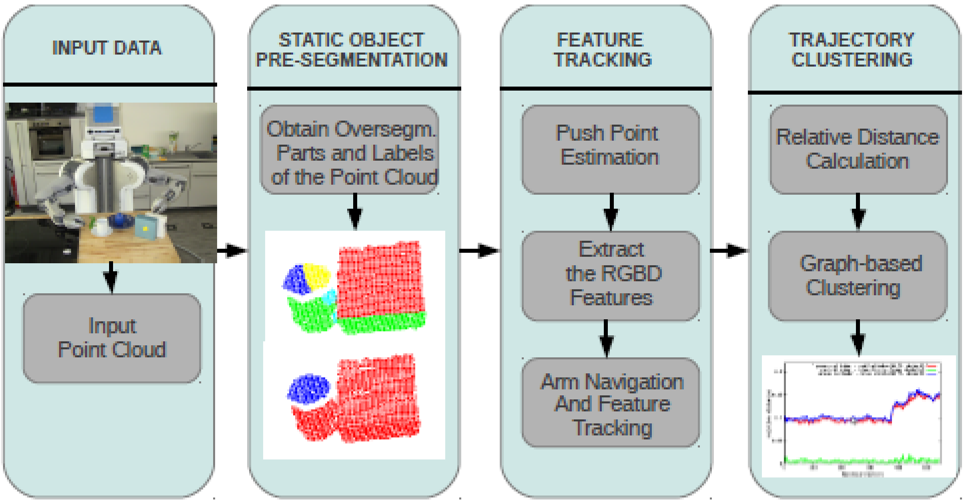

INPUT DATA
Input Point Cloud
STATIC OBJECT PRE-SEGMENTATION
Obtain Oversegm. Parts and Labels of the Point Cloud
FEATURE TRACKING
Push Point Estimation
Extract the RGBD Features
Arm Navigation And Feature Tracking
TRAJECTORY CLUSTERING
Relative Distance Calculation
Graph-based Clustering

Fig. 12. System pipeline.

After the object hypotheses O had been obtained, it was inferred, in the second node, which hypothesis was the most uncertain for being segmented correctly. For that one has to count the number of parts the respective object hypotheses O consists of and then sample from the Poisson distribution presented before. After obtaining the wrongly segmented hypotheses one has to iteratively find the contact points and the push directions as proposed in Sec. IV. It is crucial to use categorization of the objects as a prior for tracking. Afterwards, one has to extract and track line and corner features on the flat object hypotheses and circle and cylinder features on the round ones. Finally, the arm motion movement in 1cm intervals is executed until one or more objects were successfully clustered or a maximum 10 number of pushes is reached. All of the features are being tracked during the interaction and the trajectories of feature centroids are being saved. In sixth row of Fig. 11 the relative distances between centroids over the time are depicted. Based on relative distances, 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.

C. Results

Fig. 13 shows an evaluation of the clustering algorithm on 11 scenes, three of which are presented in greater detail in Fig. 11. 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 1.5 cm are recommended, and the possible under-segmentations solved by applying the whole method iteratively until all the objects are clearly separated. One of the scenes used in the experiment for Fig. 13 was solely responsible for not reaching overall 100% success rate in the 0.016-0.02 cm range for $d_{threshold}$, as the two objects therein could not be separated. There the objects rotated roughly around their centers, and in this case the method treats them as if it was a single (articulated) object. It is planned to alleviate this by integrating the rotations and improving upon the accuracy of the tracker. For the scene in column 3 of Fig. 11 it can be observed that there is only one feature on each object which results in one feature distance plot only. All the clustering algorithms trying to explicitly cluster at least one pair of features with the constant relative distance over the time would fail in this use.

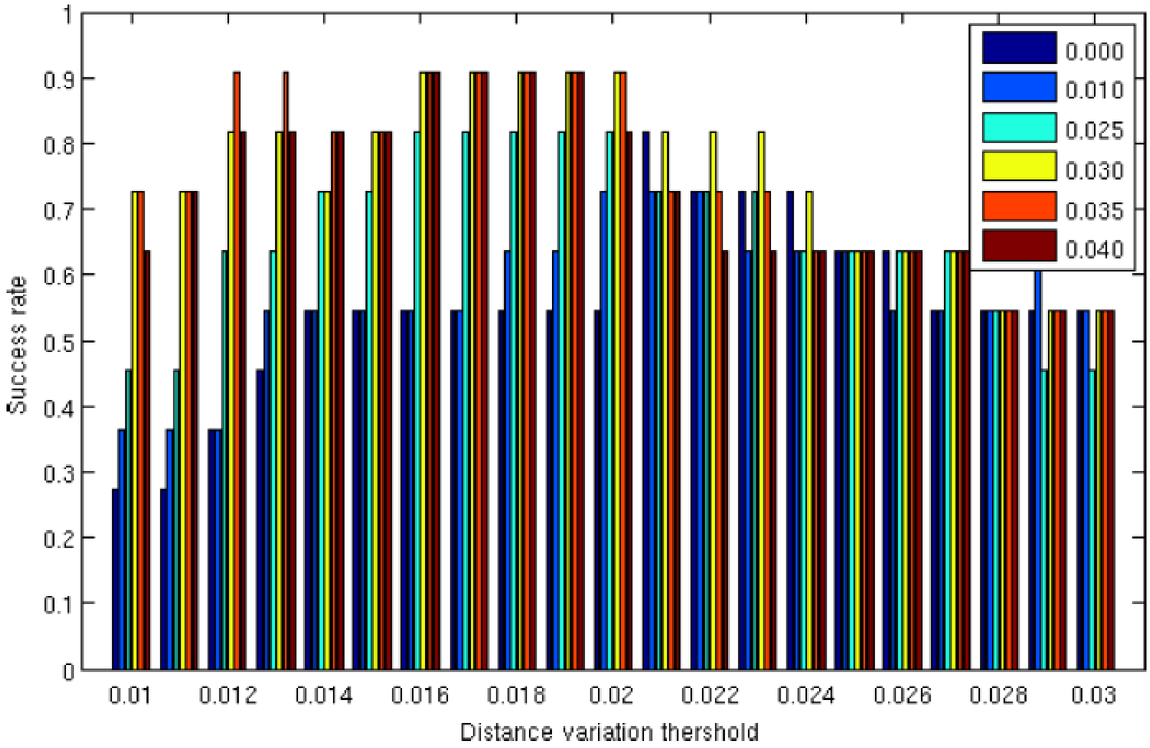


Fig. 13. Clustering success rate on 11 scenes for different values of $p_{threshold}$ (see legend) as a function of $d_{threshold}$.

Using the graph-based clustering method it is possible to disconnect the two nodes of the graph and infer that these two features belong to two different objects.

D. Discussion

Though the preliminary results of the herein presented system are very promising, there is still several improvements to be made. First, the tracking needs to be made in a more accurate way because Fig. 5 row 6 suggests that the tracker currently performs with an accuracy of up to 1cm. Secondly, it is planned to combine orientations with the translations in the clustering algorithm in order to support rotational movements around the feature centroids. The final outcome of the system should be the point cloud where objects are separated based on the features that were tracked. Full model reconstruction will thus be the last step to be concentrated on in the future. In order to achieve this the system will be meshed with the region growing algorithms with the seed points being the features assigned to the objects. To cope with the occlusions multi-view segmentation approaches will be explored. Lastly, to alleviate the contact point and push direction heuristic it is planned to make use of the physics-based simulator and learn the best push policy.

XII. CONCLUSIONS AND FUTURE WORK

In this thesis an interactive perception system suitable for the segmentation of objects in cluttered scenes for the purpose of mobile manipulation was presented. 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. The system on a set of challenging cluttered scenes was evaluated and they were successfully segmented in more than 89% of cases. The results show applicability of the system for objects of various sizes, shapes and surface.

In the second part of the thesis a novel interactive perception system suitable for the segmentation of textureless objects in cluttered tabletop scenes was presented. Integrated in the system are the pre-static segmentation based on the geometrical categorization, a push direction for the robot's manipulator to induce distinct motions that enable the clustering, RGBD features suitable for tracking of textureless objects and the graph-based clustering algorithm. The system was tested on a set of 11 scenes and they are successfully segmented in more than 90% of cases. The results show applicability of the system for objects of similar colors, shapes and sizes on (for now) predominantly flat and round surfaces.

In the future it is planned to improve the latter and to integrate an arm motion and grasp planner [27] 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. Bergstreom, 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 mincut/ 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] 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.
- [17] 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 Proceedings of Spatial Cognition (SC 2012), Abbey Kloster Seeon, Germany, 2012.
- [18] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” Int. J. Comput. Vision, vol. 59, no. 2, pp. 167–181, 2004.
- [19] Q. Zhan, Y. Liang, and Y. Xiao, “Color-based segmentation of point clouds,” 2009.
- [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] K. Lai, L. Bo, X. Ren, and D. Fox, “A large-scale hierarchical multi-view rgb-d object dataset,” in Proc. of International Conference on Robotics and Automation (ICRA), 2011.
- [22] 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.

- [23] 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.
- [24] 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, Sep. 1977.
- [25] P. Azad, D. Munch, T. Asfour, and R. Dillmann, “6-dof model-based tracking of arbitrarily shaped 3d objects,” in ICRA, 2011.
- [26] D. Fox, “Kld-sampling: Adaptive particle filters,” in Advances in Neural Information Processing Systems 14. MIT Press, 2001.
- [27] 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.