 DOI: 10.3217/978-3-85125-663-5-01

Action Selection for Interactive Object Segmentation in Clutter

Timothy Patten, Michael Zillich and Markus Vincze

***Abstract*— Robots operating in human environments are often required to recognise, grasp and manipulate objects. Identifying the locations of objects amongst their complex surroundings is therefore an important capability. However, when environments are unstructured and cluttered, as is typical for indoor human environments, reliable and accurate object segmentation is not always possible because the scene representation is often incomplete or ambiguous. We overcome the limitations of static object segmentation by enabling a robot to directly interact with the scene with non-prehensile actions. Our method does not rely on object models to infer object existence. Rather, interaction induces scene motion and this provides an additional clue for associating observed parts to the same object. We use a probabilistic segmentation framework in order to identify segmentation uncertainty. This uncertainty is then used to guide a robot while it manipulates the scene. Our probabilistic segmentation approach recursively updates the segmentation given the motion cues and the segmentation is monitored during interaction, thus providing online feedback. Experiments performed with RGB-D data show that the additional source of information from motion enables more certain object segmentation that was otherwise ambiguous. We then show that our interaction approach based on segmentation uncertainty maintains higher quality segmentation than competing methods with increasing clutter.**

I. Introduction

Robots are becoming important for a diverse range of household applications, such as cleaning, tidying, gardening and personal care. These tasks require a robot to recognise, grasp and place items in their correct locations or use items appropriately. Distinguishing individual objects from other objects and the background is a first step for achieving these high-level actions. Making this distinction is a challenging problem, especially in human environments, because of the high variability of structure and the presence of clutter or occlusion. In these scenarios, it is difficult to accurately identify the portions of the data that belong to individual objects in a scene. Often the perceived environment is incomplete and this leads to an ambiguous interpretation. Typical segmentation approaches address these challenges by fine tuning parameters and exploiting structure (e.g., planar surfaces) or complex features. However, tuned parameters, structure and features for specific scenarios do not necessarily transfer to other scenarios. Consequently, the generality of these approaches is limited. For arbitrarily shaped objects or scenes with heavy occlusion, it is non-trivial to combine separated segments that in fact belong to a single object.

Fig. 1: Segmenting a cluttered scene with interaction. Left: Robot observing the scene. Middle: Robot inducing object motion. Right: Segmentation after interaction. Moved object (top left in blue) is correctly segmented.

In this paper we enable a robot to directly *interact* with the objects in a scene in order to resolve object ambiguity. The object motion induced by the robot is an additional source of information that is used in the segmentation process. Previous work on interactive segmentation apply only fixed motions; focus on segmenting an object from the background; or do not segment during interaction, only after each action is complete. Our work addresses these issues to enable interactive segmentation in real-world scenarios. First, we make use of a probabilistic segmentation approach to provide cues for where and how the robot should interact given the full state space of the manipulator, which might require complex motion plans. The scene is analysed and directly exposes the most uncertain regions where interaction is likely to improve object segmentation. Second, a combination of dense optical flow and sparse feature tracking is applied to monitor objects during interaction and to recover the motion of occluded objects (e.g., by the manipulator itself). This enables the identification of the individual motions of multiple objects with respect to each other and the background. Lastly, statistics about the pushed region of the scene are maintained during the interaction. The robot receives online feedback and updates its actions according to the segmentation quality at each time instance.

Experiments are performed with RGB-D data of cluttered scenes to first demonstrate the probabilistic segmentation process with moving objects. We benchmark against existing work of [22] and show that not only does interaction singulate objects and simplify object segmentation, but incorporating motion cues into the segmentation process leads to a significant performance boost. The second set of experiments compare the proposed active interaction strategy with competing methods. The results show that our approach maintains higher quality segmentation in comparison to the other methods as the amount of clutter increases.

*This work was supported by the European Community's Seventh Framework Programme under grant agreement No. 610532 SQUIRREL.

TP and MV are with the Vision for Robotics Laboratory, Automation and Control Institute, TU Wien, 1040 Vienna, Austria {patten, vincze}@acin.tuwien.ac.at

MZ is with Blue Danube Robotics, Vienna, Austria zillich@bluedanuberobotics.com

The remainder of this paper is organised as follows. Sec. II reviews related work. Sec. III provides an overview of the system. Sec. IV describes the probabilistic segmentation algorithm and Sec. V develops the active interactive segmentation approach. Sec. VI presents results for experiments with RGB-D data. Finally, the paper is concluded in Sec. VII with an outlook of future work.

II. Related Work

Interactive perception is the problem of interacting with the environment to improve perception quality and has been applied to many tasks such as recognition, modelling and pose estimation [2]. Interactive segmentation focuses on applying non-prehensile actions with a mobile manipulator to improve the separation of visual data into individual entities, i.e., segments. Pioneering work demonstrated the principle of interactive segmentation [10], [20], [16], which has been extended through numerous innovations. Many methods, however, are passive with respect to the segmentation task as actions are pre-planned according to expected object kinematics [15], [19] or any action expected to move an object hypothesis is chosen [17], [25], [23], [5], [11]. Accordingly, segmentation is only improved as a coincidence and actions are not chosen to directly segment objects in clutter.

Active approaches to interactive segmentation, on the other hand, specifically select actions that are expected to disambiguate a scene. A common technique is to apply actions that physically separate or "singulate" objects from a cluttered pile. This has been addressed with heuristics based on avoiding other local object clusters [4], spreading objects through orthogonal motions [12], the splitting plane between object hypotheses [14] or observed corners with local concavities [1], [13]. An alternative approach is presented by Eitel at al. [7], who use a neural network to learn favourable push actions. In contrast to these methods, we select push actions based on local segmentation uncertainty to most quickly resolve ambiguity. Resultingly, our method is driven by the observed data and does not rely on handcrafted heuristics or training data.

Our approach to interactive segmentation exploits a probabilistic representation and an information-theoretic measure for action selection. This is similar in principle to the method of van Hoof et al. [26], who maintain a probability distribution over segmentation and select actions based on expected information gain. The main difference of our method is that we continuously monitor the interaction online and use the motion sequence instead of comparing the scene immediately before and after a push. An advantage of continuous tracking is that information about the trajectory information can be used to update the segmentation. Xu et al. [27] use information theory to select actions and also track segment hypotheses during interaction. However, this information directly fuses local regions into objects where as we apply Bayesian updates to the probabilistic edges of the graph. We maintain patch-wise relations over multiple interactions, which allows the system to recover from ambiguous observed motion that can sometimes result in over-segmentation.

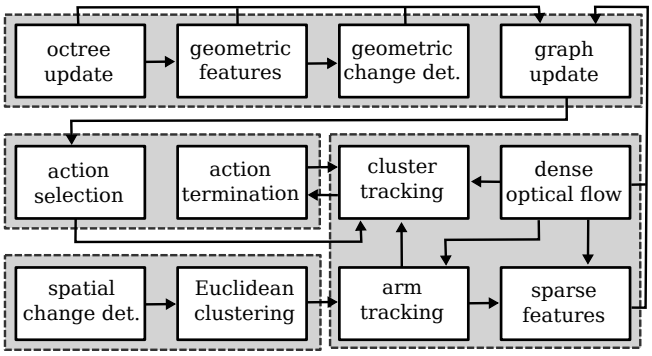

Fig. 2: System overview.

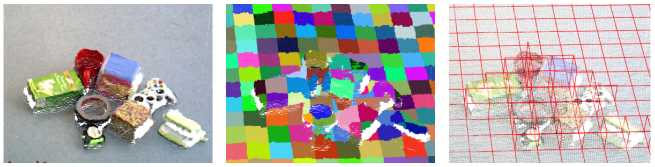

Fig. 3: Data structures for segmentation. Left: RGB-D input. Middle: Points binned in octree (coloured voxels). Right: Neighbourhood graph with edges connecting adjacent voxels.

III. System Overview

An overview of our system is illustrated in Fig. 2. The input point clouds are generated from an RGB-D sensor. The point clouds are stored in an octree, which is used to construct a neighbourhood graph. The edges in the graph represent the similarity between components of the scene and probability values are determined based on the similarity of geometric features. The edge probabilities are updated when objects are moved by exploiting dense optical flow and sparse feature tracking as well as geometric change detection as outlined in Sec. IV. The movement of objects is induced by the robot by selecting actions to best resolve segmentation ambiguity that is determined from the uncertainty in the neighbourhood graph as described in Sec. V. Online, during the action, the robot arm is identified and tracked to remove it from the segmentation and to account for occlusion. The scene, represented as clusters of nodes in the graph, is also monitored. The observed motion provides feedback that determines when the action is terminated. Finally, the process repeats by planning the next action with the updated neighbourhood graph.

IV. Probabilistic Object Segmentation

This section outlines the probabilistic object segmentation framework. Point clouds are stored in an octree and a neighbourhood graph is constructed. The probability of nodes belonging to the same object are computed according to the similarity of geometric features. When the scene is observed to move, optical flow and change detection are used to update edge probabilities.

A. Neighbourhood Graph

Each RGB-D input z_t at time t is stored in an octree as shown in Fig. 3 (middle). The neighbourhood graph $G =$

$(\mathscr{V},\mathscr{E})$ is initialised by generating graph nodes $i \in \mathscr{V}$ for every voxel. Edges $e(i,j) \in \mathscr{E}$ are generated between adjacent nodes i and j. An example of a neighbourhood graph is shown in Fig. 3 (right).

Each node i stores a collection of points as well as the centroid $\mathbf{p}^i \subseteq \mathbb{R}^3$, average normal vector $\mathbf{n}^i \subseteq \mathbb{R}^3$ and curvature $c^i \subseteq \mathbb{R}$ from the most recent RGB-D input. Edges in the graph encode the similarity between connected parts. The existence of an edge encodes the fact that the two parts are adjacent. In addition, a weight is assigned to each edge that measures their geometric similarity according to

$$g_t^{ij} = f_p(\mathbf{p}_t^i, \mathbf{p}_t^j) f_n(\mathbf{n}_t^i, \mathbf{n}_t^j) f_c(c_t^i, c_t^j). \quad (1)$$

The weight is the product of the independent probabilities given by the geometric properties of the nodes. We model the probabilities with the exponential functions

$$f_p(\mathbf{p}_t^i, \mathbf{p}_t^j) = \exp\left(-\frac{||\mathbf{p}_t^i - \mathbf{p}_t^j||^2}{2\sigma_p^2}\right), \quad (2)$$

$$f_n(\mathbf{n}_t^i, \mathbf{n}_t^j) = \exp\left(-\frac{\left(\cos^{-1}(\mathbf{n}_t^i \mathbf{n}_t^j)\right)^2}{2\sigma_n^2}\right), \quad (3)$$

$$f_c(c_t^i, c_t^j) = \exp\left(-\frac{(c_t^i - c_t^j)^2}{2\sigma_c^2}\right), \quad (4)$$

for the Euclidean distance between point centroids (2), angular difference between normals (3) and scalar difference between curvatures (4). These features capture spatial compactness and smoothness. The constants σ_p, σ_n and σ_c scale the values. In our experiments, we set σ_p to 0.85 times the voxel resolution, σ_n to 0.45π and σ_c to 0.5.

Edge relations between small patches cannot comprehensively identify similarity, especially for objects with sharp edges or irregular shape. Therefore, we combine learned features to overcome problems of locality. To achieve this, the method of [22] is performed on the original input. This procedure generates a list of geometrically consistent patches $\mathscr{P}$ from a pre-segmentation stage and computes pairwise relations based on their features. Patches are grouped by performing a graph cut. This information is mapped to the neighbourhood graph G to compute the final edge weight

$$w_t^{ij} = \alpha g_t^{ij} + (1-\alpha) l_t^{ij}, \quad (5)$$

where $l_t^{ij} = 0.9$ if i and j belong to the same patch in $\mathscr{P}$ or 0.1 otherwise. In our experiments, we set $\alpha = 0.5$.

Single input methods generate the most likely segmentation from the observed scene. We instead improve upon the initial segmentation by accumulating more data overtime. The graph is preserved and the edge weights are updated by inducing and observing motion.

B. Dense Optical Flow and Sparse Feature Tracking

The motion of the scene is monitored using a combination of dense optical flow and sparse feature tracking. Dense optical flow is determined for an RGB image by tracking pixel gradients using the method of Farnebäck et al. [8]. This assigns a flow vector to each pixel that is then projected back to the point cloud. Regions occluded by the arm are tracked by computing a Lucas Kanade (LK) [18] feature for the corresponding pixels. When interaction is terminated and the manipulator is removed from the scene, the sparse features are matched to pixels in the final RGB image using the pyramidal implementation of the LK feature tracker [3]. Additionally, if dense optical flow fails at any point during interaction, LK features are computed from the previous frame and added to the set of sparse features. These are also matched to the scene when interaction stops. This addition of tracking sparse points assists the overall flow estimation because it is more robust to occlusions. It enables flow to be computed for portions of the scene that become temporarily occluded, for example, by the manipulator or other objects. An example of dense optical flow for an intermediate frame during interaction is shown in Fig. 4 (bottom row, third from left). In Fig. 4 (bottom row, right) the image pixels are coloured to distinguish the type of motion detection. Red indicates dense optical flow, yellow indicates tracked LK features and blue indicates the static scene (flow with magnitude less than the octree resolution).

After interaction, the octree is updated with the final point cloud and graph nodes are regenerated. All pixels that are tracked during motion are used to estimate the flow for the graph nodes. For each voxel represented by a graph node, the voxel motion vector $\mathbf{m}_t^i \subseteq \mathbb{R}^3$ is computed by averaging the flow vectors from each point that belonged to the voxel prior to interaction. The new location of the voxel is computed according to

$$\mathbf{p}_{\text{new}}^i = \mathbf{p}_{t-1}^i + \mathbf{m}_t^i. \quad (6)$$

The identity i at time t is determined as the voxel in $\mathscr{V}$ nearest to $\mathbf{p}_{\text{new}}^i$.

In some cases, flow may have been corrupted and the previous identity of a voxel cannot be found. These voxels initialise nodes with no history.

C. Change Detection

Further robustness is achieved by detecting change in the scene. In most cases, optical flow suffices to estimate object motion. However, under strong occlusion, where pixels never reappear, no history can be determined. Therefore, we also compute the change for voxels (and neighbours of these voxels) that do not have a valid flow vector assigned. Change detection is performed at the end of interaction by comparing the new values for the point centroid, normal and curvature with the previous values. The magnitude of change for node i at time t is given by

$$\Delta(i, z_t) = 1 - f_p(\mathbf{p}_t^i, \mathbf{p}_{t-1}^i) f_n(\mathbf{n}_t^i, \mathbf{n}_{t-1}^i) f_c(c_t^i, c_{t-1}^i), \quad (7)$$

where $0 \leq \Delta(i, z_t) \leq 1 \ \forall i, t$.

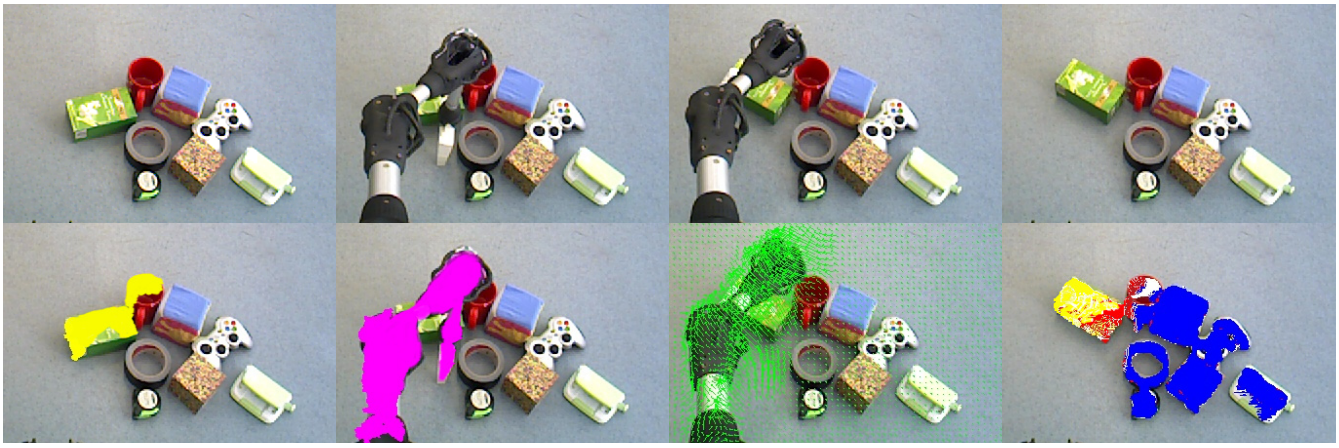

Fig. 4: Example of motion analysis. Top: Initial scene, introduction of arm, snapshot of arm moving an object and arm removed before next interaction. Bottom: Occluded regions on objects that initialise the set of LK features, arm identified with spatial change detection, dense optical flow field and motion detection (successful dense optical flow in red, static in blue, tracked LK features in yellow, invalid flow in white and background/ground unmodified).

D. Recursive Object Segmentation

The flow vectors computed in Sec. IV-B determine the previous identity of each graph node at the previous time instance (i.e., before the interaction). Correspondingly, neighbouring nodes that were also neighbours before the interaction will maintain their edge. This probability is updated by Bayes' rule where the new similarity probability is the likelihood and the previous probability is the prior.

Consider an edge $e(i,j)$ between two nodes i and j at time t. Let the continuous random variable x^{ij} denote the state between two nodes in the graph i and j. The probability of two nodes belonging to the same object given the RGB-D measurements is expressed by the conditional probability $p(x^{ij}|\mathbf{z}_{0:t})$ where $\mathbf{z}_{0:t} = \{\mathbf{z}_0, \ldots, \mathbf{z}_t\}$ is the history of measurements. If these nodes have valid motion vectors and if the edge exists at time $t-1$, then the nodes were also neighbours. In this case, $p(x^{ij}|\mathbf{z}_{0:t-1})$ is known, otherwise, the nodes were not neighbours and the similarity probability is unknown, therefore a uniform prior is assigned, i.e., $p(x^{ij}|\mathbf{z}_{0:t-1}) = 0.5$. The probability of any edge $e(i,j)$ between nodes i and j at time t given the *history* of measurements can be computed according to Bayes' rule

$$p(x^{ij}|\mathbf{z}_{0:t}) = \frac{p(x^{ij}|\mathbf{z}_{0:t-1})p(\mathbf{z}_t|x^{ij})}{p(\mathbf{z}_t|\mathbf{z}_{0:t-1})}, \tag{8}$$

where $p(x^{ij}|\mathbf{z}_{0:t-1})$ is the prior, $p(\mathbf{z}_t|x^{ij}) = w_t^{ij}$ is the measurement likelihood and the denominator is given by

$$p(\mathbf{z}_t|\mathbf{z}_{0:t-1}) = \int p(\mathbf{z}_t|x^{ij})p(x^{ij}|\mathbf{z}_{0:t-1})dx. \tag{9}$$

The posterior maintains the total probability after many point cloud inputs. Consequently, neighbouring nodes that move along similar trajectories will recursively update their similarity probability. On the other hand, nodes that become neighbours but have no history are initialised with the similarity probability of the most recent observation due to the uniform prior.

The Bayesian updates require the integral in (9) to be computed. To keep the updates tractable, we compute the denominator with

$$\begin{aligned} p(\mathbf{z}_t|\mathbf{z}_{0:t-1}) \quad \approx \quad & p(\mathbf{z}_t|x^{ij})p(x^{ij}|\mathbf{z}_{0:t-1}) \; + \\ & p(\mathbf{z}_t|\neg x^{ij})p(\neg x^{ij}|\mathbf{z}_{0:t-1}), \end{aligned} \tag{10}$$

where $p(\mathbf{z}_t|\neg x^{ij}) = 1 - p(\mathbf{z}_t|x^{ij})$ and $p(\neg x^{ij}|\mathbf{z}_{0:t-1}) = 1 - p(x^{ij}|\mathbf{z}_{0:t-1})$. This approximates the random variable to a discrete domain with two possible states x^{ij} and $\neg x^{ij}$.

Edges between nodes without a valid flow vector are updated using the change detection. The prior probability is computed as

$$p(x^{ij}|\mathbf{z}_{0:t-1}) = \min\left(0.5, \min(\Delta(i,z_t), \Delta(j,z_t))\right). \tag{11}$$

When the magnitude of the change for both nodes is greater than 0.5, the prior is set to the minimum change value and the resulting posterior $p(x^{ij}|\mathbf{z}_{0:t}) > p(\mathbf{z}_t|x^{ij})$. The increase of the posterior from the likelihood captures the fact that the nodes likely moved but the optical flow failed. If neither or both change magnitudes are less than 0.5, the prior is set to a uniform probability, $p(x^{ij}|\mathbf{z}_{0:t-1}) = 0.5$. Applying (8) gives $p(x^{ij}|\mathbf{z}_{0:t}) = p(\mathbf{z}_t|x^{ij})$. Here, the magnitude of the change is not sufficient to increase the similarity probability.

V. Active Interactive Object Segmentation

This section describes the active interaction approach for improving object segmentation. The uncertainty from the probabilistic segmentation is exploited to generate contact points and directions where interaction is considered to be most useful. During interaction the scene is monitored. The robot gathers evidence in real time to determine if the consequence of the planned action can assist the segmentation. When sufficient data is collected, the action is terminated so that the process can repeat.

A. Action Determination

Determining the best push action first generates a set of candidate pushes, then evaluates the quality of each push and finally selects the push with the highest quality. Unlike other methods that sample the action space, e.g., [26], our set of candidate push actions is generated by the geometry of the scene and it captures all possible stable actions. Evaluating the actions exploits the uncertainty from the probabilistic segmentation. The uncertainty provides very informative cues about which regions of the scene are most ambiguous and therefore which regions would most benefit from interaction.

1) Action Candidate Generation: Neighbouring nodes are clustered if their edges have high similarity probability as well as if the nodes have similar normal vectors. Merging nodes with high similarity attempts to group nodes into reasonable object hypotheses. Enforcing nodes to have similar normal vectors results in clusters with smooth surfaces, which are more easily pushed than irregular surfaces.

The contact point for each cluster is computed as the median of all points belonging to the nodes in the cluster. Each candidate generates two potential push vectors in the orthogonal directions to the normal of the surface represented by the cluster. The initial set of push candidates is refined by removing pushes that are perpendicular to the support plane (e.g., floor or table). These push directions imply pushing into or away from the fixed surface. Furthermore, candidates are removed if they are not accessible by the manipulator. This occurs if a collision free path to the contact point cannot be found or if the contact point is beyond the reach of the manipulator. The set of candidate push actions is denoted $\mathscr{A}$.

2) Action Evaluation: Every candidate action $\mathbf{a} \in \mathscr{A}$ is evaluated by computing a score

$$s(\mathbf{a}) = \frac{\bar{H}}{d_c + d_\theta + d_m + N_c}. \tag{12}$$

This assigns a high score to candidates with high entropy (uncertainty) $\bar{H}$ and divides the values by the distance to the circumference of the clutter pile (d_c), the length of the chord from the contact point to the clutter pile circumference (d_θ), the distance to the end effector (d_m) and the number of points in the point cloud in the immediate vicinity of the cluster along the push direction (N_c). The entropy $\bar{H}$ represents the uncertainty of the surrounding segmentation of the cluster and it is computed by averaging the entropy of the edges from every node within the cluster to every node not in the cluster. The distance from the push contact point to the circumference of the clutter pile d_c and the length of the chord from the contact point to the clutter pile circumference d_θ favour actions that are more likely to singulate an object. Intuitively, an object is pushed further away from other objects if it is near to the circumference of the clutter pile and the push direction is short, i.e., towards the circumference and away from the centre of the clutter. The distance to the end effector d_m favours clusters that are nearer for manipulation and therefore require less motion. The number of points N_c represents the likelihood of colliding the cluster into another object. This would be counter productive for segmentation because objects should ideally be pushed individually.

3) Action Selection: The best action is selected as the push with the highest score. Formally, the selected action is given by

$$\mathbf{a}^* = \arg\max_{\mathbf{a} \in \mathscr{A}} s(\mathbf{a}). \tag{13}$$

B. Action Termination

During interaction the robot monitors the selected cluster. The robot reasons about the impact the induced motion will have on the subsequent segmentation quality and decides to terminate the action if there is sufficient information to confidently separate or merge parts of the scene.

Denote the set of neighbours to cluster $\mathbf{c}$ as $\mathscr{N}(\mathbf{c})$. During the push, optical flow is computed for every pixel of $\mathbf{c}$ and $\mathscr{N}(\mathbf{c})$. The optical flow is projected back to the point cloud to generate a motion vector for each point in 3D and the distance moved by each point is calculated. At each time t, the average distance from the start location for every point in the cluster is computed, denoted $\bar{d}_t^{\mathbf{c}}$. Then for each neighbouring node $n \in \mathscr{N}(\mathbf{c})$ we compute the following

$$\sigma_t^n = \sqrt{\frac{\sum_{k=1}^{N^n} (d_t^{nk} - \bar{d}_t^{\mathbf{c}})^2}{N^n - 1}}, \tag{14}$$

where d_t^{nk} is the distance traveled by point k of neighbour n that has total N^n points. Intuitively, this quantity is similar to the standard deviation, but it is computed with the mean value of the cluster. For this case, σ_t^n quantifies the amount of variation with respect to the cluster. A low value means that the distance travelled by neighbouring points is close to the mean distance of the cluster. Consequently, the neighbour has moved in a similar way as the cluster that was pushed and the neighbour is likely to belong to the cluster. On the other hand, a high value means that the distance travelled by neighbouring points is spread over a wide range of values and differ greatly to the average distance travelled by the cluster. In this case, a significant portion of the neighbour has moved differently to the pushed cluster and the components of the neighbour likely belong to different objects.

Furthermore, the pushed cluster is monitored and the variation of the distance of its points is measured with

$$\sigma_t^{\mathbf{c}} = \sqrt{\frac{\sum_{k=1}^{N^{\mathbf{c}}} (d_t^{\mathbf{c}k} - \bar{d}_t^{\mathbf{c}})^2}{N^{\mathbf{c}} - 1}}, \tag{15}$$

where $d_t^{\mathbf{c}k}$ is the travelled distance of point k with $N^{\mathbf{c}}$ points in total. This is precisely the standard deviation: High variation suggests that some parts of the cluster moved differently, therefore, the parts may belong to different objects.

This gives rise to three termination criteria

$$\forall n \in \mathscr{N}(\mathbf{c}) \qquad \bar{d}_{\mathbf{c}}^t > \delta_1 \wedge \sigma_t^n < \delta_2, \tag{16}$$

$$\exists n \in \mathscr{N}(\mathbf{c}) \quad \text{s.t.} \quad \sigma_t^n > \delta_3, \tag{17}$$

$$\sigma_t^{\mathbf{c}} > \delta_4. \tag{18}$$

The first criterion (16) means that the cluster has moved a significant distance δ_1 and all neighbours have moved a similar distance determined by the threshold δ_2. The second criterion (17) means that there is at least one neighbour that has moved in a different manner to the cluster as determined by the threshold δ_3. The third criterion (18) means that the cluster itself has accumulated a high amount of variation as determined by the threshold δ_4. In our experiments, we set the values to δ_1 to four times the voxel resolution and d_2, d_3, and d_4 are set to 0.5.

C. Arm Tracking

A major limitation of previous interactive segmentation work is the confinement of studying pairs of RGB-D frames or RGB images, where an observation is made before and after individual fixed-length pushes. The reason is because of the degradation of feature tracking algorithms caused by occlusion created by the tool that moves the objects.

To increase robustness to the occlusion, the manipulator of the robot is explicitly tracked during interaction. Before objects are pushed, the manipulator is expected to be the only moving object in the scene. The manipulator is identified by using spatial change detection in 3D. Euclidean clustering is then performed and the points belonging to the largest cluster are considered the manipulator, smaller clusters are considered noise. Each point of the manipulator is continuously tracked with dense optical flow and removed from the input to avoid association of the arm to the scene. Furthermore, the region of the scene occluded by the manipulator is identified. The corresponding nodes initialise the set of sparse features that are matched to the scene at the end of interaction once the manipulator moves to a new position. Details of dense and sparse tracking follows the procedure in Sec. IV-B.

An example of identifying the manipulator and tracking occluded points is shown in Fig. 4. The manipulator is detected (bottom row, second from left) and the occluded voxels are determined in the initial frame (bottom row, left) then matched to points in the final frame (bottom row, right).

Complex or fast motion may cause tracking failure. However, this method is sufficient for relatively short and slow push actions. More sophisticated manipulator tracking and removal, e.g., [6], could be implemented if necessary.

VI. Experiments

This section presents results from experiments with cluttered scenes of objects. The first set of experiments demonstrates the recursive probabilistic segmentation method to correctly identify the ambiguous regions and compares the segmentation accuracy against a static single input method. The second set of experiments showcases the interaction planning strategy by comparing performance against competing methods.

A. Metrics and Ground Truth Generation

In order to evaluate our interactive segmentation approach, the most likely segmentation is generated by performing a graph cut on the neighbourhood graph [9]. For all experiments, we set the cut threshold to 0.5. Segmentation performance is measured by adapting the method in [21] for point clouds. Each object from the segmented point cloud is compared to the ground truth labelled objects. The largest overlapping object is assigned to each ground truth object. The precision is computed as the fraction of points assigned to the ground truth object and the recall is the fraction of points from the ground truth assigned to the object. Precision and recall are combined to compute the F1 score, $F1 = 2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$, which is the harmonic mean between the two quantities.

Ground truth labelling is computed at the beginning and end of each trial, no ground truth can be acquired for the intermediate stages when multiple interactions are applied. Ground truth segmentation is generated using the method of [24]. We place (or remove, for the end scene) one object at a time and record a point cloud. Scene differencing on the depth data is run at each stage of the sequence to detect the introduction (or loss) of an object. These points are assigned a unique label. The final scene has one label for the background and a unique label for each object.

B. Recursive Segmentation

The first set of experiments analyses our interactive segmentation approach in comparison to single frame segmentation. As a benchmark, we choose the method of [22], which will be referred to as *static* segmentation. A total of 10 cluttered scenes with two to eight objects were randomly generated. Our interactive segmentation approach applied a random number of pushes for each scene before terminating.

The precision, recall and F1 score for all object instances are shown in Fig. 5. For interactive segmentation, the result is computed from the graph cut output on the neighbourhood graph after the final interaction. For static segmentation, the results are generated by directly segmenting either the final or initial point clouds. Fig. 5 shows that interactive segmentation achieves a higher accuracy than static segmentation and that the final scene is segmented more accurately than the initial scene on average. This outcome highlights two important aspects about interactive segmentation. Firstly, intelligent induced motion improves accuracy for static segmentation approaches. This is often due to the clutter being dispersed and objects becoming singulated after being pushed. Secondly, although the final scene is more easily segmented, accumulating the information during a sequence of interactions leads to an even further boost in performance.

In Fig. 6 we plot the difference in F1 score between interactive and static segmentation on the final scene for each trial. The F1 scores are computed by averaging the scores of each object in the scene. This shows that most trials $(8/10)$ are segmented better when motion information is incorporated. In two trials, interactive segmentation performs worse than the benchmark. This can occur sometimes when multiple large portions of the scene move together and objects become under-segmented.

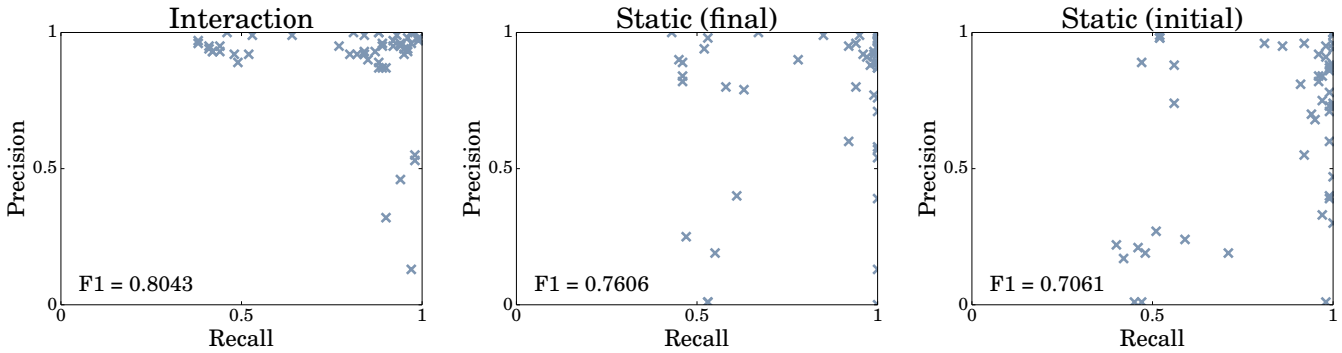

Fig. 5: Precision and recall of all objects for 10 sets of randomly cluttered scenes of varying number of objects and applied interactions. F1 score shown in bottom left corner. Left: Interactive segmentation. Middle: Static segmentation performed on the final scene (after all interactions). Right: Static segmentation performed on the initial scene (before any interaction).

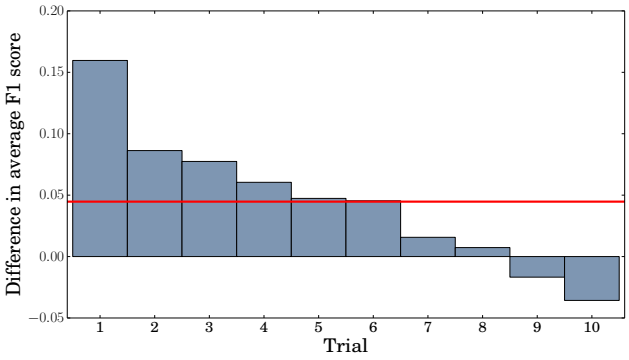

Fig. 6: Difference between segmentation accuracy for recursive over static method for 10 trials (sorted highest to lowest). Results for each trial is computed by averaging the F1 scores of the individual objects in the scene. Average improvement over all trials shown in red.

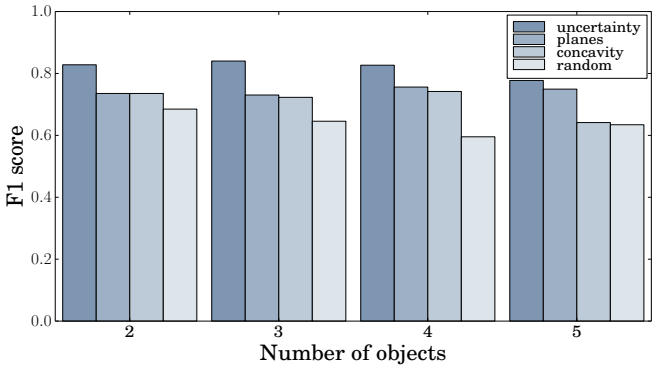

Fig. 7: Segmentation accuracy for different action selection strategies with increasing number of objects (clutter). Each bar represents F1 score by averaging three separate trials.

C. Action Selection

The second set of experiments evaluates the action selection strategy in comparison to other methods. In particular, comparisons are made with our implementations of pushing into concave corners [13], pushing along splitting planes [14] and random action selection. In these experiments, three interactions are made for each trial. Three trials are performed for each approach with sets of objects increasing in number from two to five. The segmentation accuracy for all trials is summarised in Fig. 7.

The results show that segmentation accuracy declines for all methods as clutter is increased (more objects are introduced). However, the segmentation accuracy using uncertainty to select actions declines at a slower rate. Segmentation accuracy when applying actions that attempt to split planes also remains almost constant but with lower accuracy than using uncertainty. Pushing into concavities performs well with few objects but suffers a significant performance decrease when the number of objects increases. The random strategy performs worst overall.

Intuitively, the methods of using uncertainty or splitting planes tend to push flatter surfaces and therefore often push a single object. Pushing into concavities does not consider the object hypotheses in the scene, consequently pushes tend to slip between objects. This often leads to singulation when the number of objects is small, but often leads to subsequent collisions when the number of objects is large. The collisions lead to difficult motions that result in incorrect segmentation decisions, hence the decrease in performance with five objects.

An example of action selection by each method is shown in Fig. 8. The push proposed by disambiguating uncertainty (white) targets a corner of a box. Corners can often be ambiguous because the local structure lacks smoothness. This is confirmed in the right of Fig. 8, which shows the corresponding segmentation uncertainty, computed by averaging the entropy of the edges for each node in the neighbourhood graph. Pushing into concave corners (cyan) often separates touching objects but this can sometimes be a wasteful action when objects are already easily segmented. Pushing along splitting planes (magenta) tries to confirm if object hypotheses on the opposite side of a visual edge belong together without considering the underlying segmentation algorithm. Our method, on the other hand, directly probes the scene where it is most ambiguous.

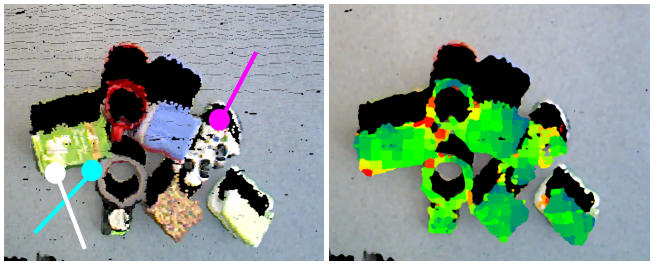

Fig. 8: Left: Example of the push actions chosen by the different interactive segmentation strategies. *Uncertainty* shown in white, *splitting planes* shown in magenta and *concavities* shown in cyan. Right: Heat map (green to red) of entropy in the neighbourhood graph, computed by averaging the entropy of the edges extending from each node.

VII. Conclusion

We have presented a method for improving object segmentation in clutter by directly interacting with the scene. Our approach identifies the uncertainty of object segmentation and selects actions that directly probe these regions in order to resolve the ambiguity. Using our proposed active approach, we are able to improve segmentation accuracy in comparison to a static segmentation method as well as maintain higher quality segmentation in cluttered scenes in comparison to competing action selection strategies.

One limitation of our work is that the camera must remain fixed. We intend to address this by incorporating SLAM or visual odometry to allow the robot to move with its base. This will permit a larger range of actions to be applied as well as more information to be gathered from new view points. We also plan to extend this to the problem of clearing piles of clutter, whereby confident object segmentation from interaction will be an important behaviour to enable successful recognition, pose estimation and grasping.

Acknowledgements

The authors thank Markus Suchi and Kiru Park for helping with generating ground truth data.

References

[1] C. Bersch, D. Pangercic, S. Osentoski, K. Hausman, Z. C. Marton, R. Ueda, K. Okada, and M. Beetz, "Segmentation of textured and textureless objects through interactive perception," in *Proc. of RSS: Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, 2012, pp. 1–8.

[2] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, "Interactive perception: Leveraging action in perception and perception in action," *IEEE Trans. Robot.*, vol. 33, no. 6, pp. 1273–1291, 2017.

[3] J. Y. Bouguet, "Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm," *Intel Corporation Microprocessor Research Labs*, 2000.

[4] L. Chang, J. R. Smith, and D. Fox, "Interactive singulation of objects from a pile," in *Proc. of IEEE ICRA*, 2012, pp. 3875–3882.

[5] K. Chaudhary, C. W. Au, W. P. Chan, K. Nagahama, H. Yaguchi, K. Okada, and M. Inaba, "Retrieving unknown objects using robot in-the-loop based interactive segmentation," in *Proc. of IEEE/SICE SII*, 2016, pp. 75–80.

[6] C. G. Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg, "Probabilistic articulated real-time tracking for robot manipulation," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 577–584, 2017.

[7] A. Eitel, N. Hauff, and W. Burgard, "Learning to singulate objects using a push proposal network," in *Proc. of ISRR*, 2017, pp. 1–13.

[8] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Proc. of SCIA*, 2003, pp. 363–370.

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

[10] P. Fitzpatrick, "First contact: An active vision approach to segmentation," in *Proc. of IEEE/RSJ IROS*, 2003, pp. 2161–2166.

[11] L. K. L. Goff, G. Mukhtar, P. H. L. Fur, and S. Doncieux, "Segmenting objects through an autonomous agnostic exploration conducted by a robot," in *Proc. of IEEE IRC*, 2017, pp. 284–291.

[12] M. Gupta, J. Müller, and G. S. Sukhatme, "Using manipulation primitives for object sorting in cluttered environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 608–614, 2015.

[13] K. Hausman, F. Balint-Benczedi, D. Pangercic, Z. C. Marton, R. Üda, K. Okada, and M. Beetz, "Tracking-based interactive segmentation of textureless objects," in *Proc. of IEEE ICRA*, 2013, pp. 1122–1129.

[14] T. Hermans, J. M. Rehg, and A. Bobick, "Guided pushing for object singulation," in *Proc. of IEEE/RSJ IROS*, 2012, pp. 4783–4790.

[15] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz, "Interactive segmentation, tracking, and kinematic modeling of unknown 3D articulated objects," in *Proc. of IEEE ICRA*, 2013, pp. 5003–5010.

[16] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *Proc. of IEEE ICRA*, 2009, pp. 1377–1382.

[17] E. S. Kuzmič and A. Ude, "Object segmentation and learning through feature grouping and manipulation," in *Proc. of IEEE-RAS HUMANOIDS*, 2010, pp. 371–378.

[18] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of IJCAI*, 1981, pp. 674–679.

[19] R. M. Martín and O. Brock, "Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors," in *Proc. of IEEE/RSJ IROS*, 2014, pp. 2494–2501.

[20] G. Metta and P. Fitzpatrick, "Better vision through manipulation," *Adapt. Behav.*, vol. 11, no. 2, pp. 109–128, 2003.

[21] E. Potapova, M. Zillich, and M. Vincze, "Attention-driven segmentation of cluttered 3D scenes," in *Proc. of ICPR*, 2012, pp. 3610–3613.

[22] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," in *Proc. of IEEE/RSJ IROS*, 2012, pp. 4791–4796.

[23] D. Schiebener, A. Ude, and T. Asfour, "Physical interaction for segmentation of unknown textured and non-textured rigid objects," in *Proc. of IEEE ICRA*, 2014, pp. 4959–4966.

[24] M. Suchi, T. Patten, D. Fischinger, and M. Vincze, "Easylabel: A semi-automatic pixel-wise object annotation tool for creating robotic RGB-D datasets," in *Proc. of IEEE ICRA*, 2019, (accepted).

[25] A. Ude, D. Schiebener, N. Sugimoto, and J. Morimoto, "Integrating surface-based hypotheses and manipulation for autonomous segmentation and learning of object representations," in *Proc. of IEEE ICRA*, 2012, pp. 1709–1715.

[26] H. van Hoof, O. Kroemer, and J. Peters, "Probabilistic segmentation and targeted exploration of objects in cluttered environments," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1198–1209, 2014.

[27] K. Xu, H. Huang, Y. Shi, H. Li, P. Long, J. Caichen, W. Sun, and B. Chen, "Autoscanning for coupled scene reconstruction and proactive object analysis," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 177:1–177:14, 2015.