MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Detection, Tracking and 3D Modeling of Objects with Sparse RGB-D SLAM and Interactive Perception

Almeida, D.; Ataer-Cansizoglu, E.; Corcodel, R.

TR2019-119 October 18, 2019

Abstract

We present an interactive perception system that enables an autonomous agent to deliberately interact with its environment and produce 3D object models. Our system verifies object hypotheses through interaction and simultaneously maintains 3D SLAM maps for each rigidly moving object hypothesis in the scene. We rely on depth-based segmentation and a multigroup registration scheme to classify features into various object maps. Our main contribution lies in the employment of a novel segment classification scheme that allows the system to handle incorrect object hypotheses, common in cluttered environments due to touching objects or occlusion. We start with a single map and initiate further object maps based on the outcome of depth segment classification. For each existing map, we select a segment to interact with and execute a manipulation primitive with the goal of disturbing it. If the resulting set of depth segments has at least one segment that did not follow the dominant motion pattern of its respective map, we split the map, thus yielding updated object hypotheses. We show qualitative results with a Fetch manipulator and objects of various shapes, which showcase the viability of the method for identifying and modelling multiple objects through repeated interactions.

IEEE-RAS International Conference on Humanoid Robots

Detection, Tracking and 3D Modeling of Objects with Sparse RGB-D SLAM and Interactive Perception

Diogo Almeida[1], Esra Ataer-Cansizoglu[2] and Radu Corcodel[3]

***Abstract*— We present an interactive perception system that enables an autonomous agent to deliberately interact with its environment and produce 3D object models. Our system verifies object hypotheses through interaction and simultaneously maintains 3D SLAM maps for each rigidly moving object hypothesis in the scene. We rely on depth-based segmentation and a multi-group registration scheme to classify features into various object maps. Our main contribution lies in the employment of a novel segment classification scheme that allows the system to handle incorrect object hypotheses, common in cluttered environments due to touching objects or occlusion. We start with a single map and initiate further object maps based on the outcome of depth segment classification. For each existing map, we select a segment to interact with and execute a manipulation primitive with the goal of disturbing it. If the resulting set of depth segments has at least one segment that did not follow the dominant motion pattern of its respective map, we split the map, thus yielding updated object hypotheses. We show qualitative results with a Fetch manipulator and objects of various shapes, which showcase the viability of the method for identifying and modelling multiple objects through repeated interactions.**

I. Introduction

Robotic manipulators can exploit their ability of interacting with the environment to enrich their perception. While many perception tasks are under-constrained and ambiguous from the point of view of a passive sensor, robotic interaction with the environment can be leveraged to resolve ambiguities in the measurements. This type of interaction-guided perception is often dubbed interactive perception (IP) [1], [2]. IP has been used to improve the performance of computer vision segmentation algorithms, as the outcome of interactions such as pushing or pulling an object can confirm or negate a segmentation hypothesis. This approach has been implemented, e.g., by tracking features during interaction or by comparing the outcome of an action with a previously observed state.

In this work, we focus on an object detection and grasping task, where a robotic agent firstly interacts with a set of objects in clutter on top of a table and grasps the detected object hypotheses. Unlike related IP works, which limit their scope to the detection and tracking of objects, we leverage a Simultaneous Location and Mapping (SLAM) technique to accumulate visual information on the object hypotheses. Each hypothesis is tracked by an independent SLAM map. Due to clutter, an object hypothesis might contain more than one object. We introduce a novel map management algorithm which leverages interaction to enable our system to recursively detect these incorrect hypotheses. The corresponding maps are then *split* into two independent maps. Our contributions are threefold:

1) An algorithm for detecting and tracking several object hypotheses in the context of a feature-based SLAM framework, driven by IP.
2) A segment classification method which allows the system to *split* object hypotheses and prevents contamination between object models.
3) A proof of concept of our integrated system, which leverages the proposed methods to detect and reconstruct several objects in an unstructured environment, and uses the generated models to inform a grasp pose detector. We show qualitatively that the accumulated visual information helps the detector to provide better grasp candidates. This is achieved without the need to resort to additional cameras or changing the robot's point of view.

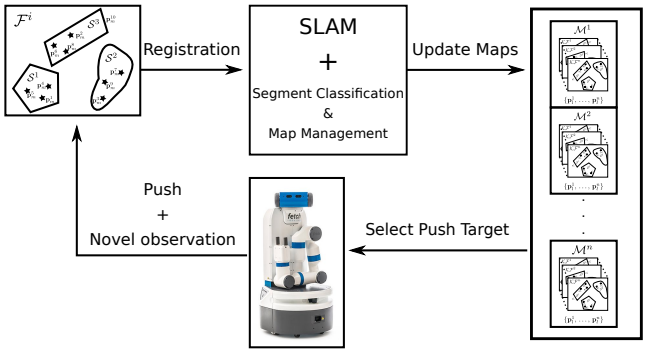

Fig. 1: Overview of our Interactive Perception system: a fetch manipulator observes a scene and obtains RGB-D keyframe observations. These are registered using sparse-SLAM agains a set of existing maps. Our proposed segment classification and map management algorithms work to *split* the maintained maps into different object hypothesis.

II. Related Work

Our main contribution is related to employing a sparse-SLAM method to drive an IP system. Thus, we firstly present related works in IP, followed by a short discussion on object tracking and reconstruction through SLAM methods.

A. Interactive Perception for segmentation

IP for improving object segmentation is a recurring theme in robotics articles. In [3], [4] a brick sorting system is proposed, in which groups of Lego bricks are segmented and an interactive algorithm is proposed, where a robot disturbs

[1]Division of Robotics, Perception and Learning, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden diogoa@kth.se

[2]Wayfair, Boston, MA 02116, USA, cansizoglu@ieee.org.

[3]Mitsubishi Electric Research Labs (MERL), Cambridge, MA 02139, USA, corcodel@merl.com

This work was realized at and supported by the Mitsubishi Electric Research Laboratories (MERL).

the groups repeatedly until bricks are singulated, and thus easily graspable. A similar pipeline is proposed in [5] for different objects, where the decision of whether an object is singulated is made through a classification of cumulative likelihood ratios.

Other methods aim at achieving an accurate scene segmentation without necessarily relying on singulation. An earlier study [6] tests object hypotheses through pushing, but results were limited to two objects. In [7], image features are tracked during robotic interaction, and a clustering algorithm is applied to group features which move with a consistent rigid-body motion, allowing for multiple objects in clutter to be detected. The method was extended to textureless objects by using depth features [8]. Katz et al. [9] track RGB-D segments from consecutive robotic interactions to grasp objects which were observed to move rigidly between two consecutive RGB-D frames. Iterative Closest Point (ICP) algorithms were also used to track object hypotheses in IP systems [10] and in [11] a probabilistic approach to the segmentation algorithm provides a robust solution to the problem.

In this work, we present an IP system which is not limited to detection and tracking, but simultaneously maintains multiple, independent maps of each detected object, through a sparse SLAM method, enabling the system to use more than the current immediate visual data to, e.g., plan for an object grasp. We show qualitatively that the accumulated visual information enables a grasp pose detection method to produce better grasp candidates.

B. Object tracking and reconstruction

The reconstruction of a scene observed by a 3D sensor, such as an RGB-D camera, is an instance of the SLAM problem, and has been addressed in many different forms. Dense methods [12]–[14] fuse several consecutive depth frames registered through ICP and are able to leverage all the available visual data to improve the reconstruction quality. In [15], a change detection method is present to enable segmentation of objects which moved in between observations of the same scene and [16] employs some interaction to disambiguate segmentation results. More recently, [17] proposes a method which is able to independently model dynamic sections of a dense map, enabling object reconstruction. However, this method struggles with small objects, such as the ones we are interested in interacting with.

Other methods [18]–[22] augment their maps with object information, obtained through diverse segmentation strategies, and employ the detected objects as map landmarks. Over several observations, changes are detected in the obtained RGB-D maps, which are used to define dynamic sections and enable the production of object models.

In our work, we employ a sparse-SLAM method [23] to track and reconstruct detected objects. We rely on interaction as the primary means of object detection and observation, that is, while adopting a static viewpoint, non-prehensile manipulation allows us to observe different views of an object and use SLAM to accumulate information. This is a significant difference w.r.t the previously mentioned methods, as they rely mostly on observing the same scene multiple times over different viewpoints, and require external disturbances on a scene to detect objects. Additionally, instead of maintaining a single map, augmented by object data, we start with a single map, which is recursively *split* into new maps as tracked features in a map fail to register after an interaction. Previous work [24] explored this idea of modelling objects in separate maps, however it relies on external interaction for object discovery and is unable to deal with contamination between maps and thus cluttered environments. We employ a classification algorithm to address this issue and prevent contamination between maps due to clutter.

III. Method

We propose to detect and track multiple objects, simultaneously and independently, by leveraging a sparse SLAM algorithm which registers points and planes in the 3D space [23] and the assumption of interaction in between observations of the environment. We will use the standard definitions of *measurements* and *landmarks* in the context of SLAM: *measurements* are extracted by the system from the available RGB-D data, and are associated to *landmarks* in a map. Each detected object is tracked in its own map, in contrast with methods that add objects to the state information in a single map [18], or segment them out of a map [15], [19].

When our system receives a new RGB-D frame, we process it to extract point and plane measurements, and perform depth-based segmentation. The measurements are used to register each frame with respect to all of the existing maps in the system. The outcome of this registration procedure is used to classify the segments and allows us to prevent contamination between maps. We split a map into two independent hypotheses, if the outcome of an interaction results in segments that do not register properly with any map.

A. Definitions

We denote our *measurements* as $\mathbf{p}_m$ and their corresponding *landmarks* are expressed as $\mathbf{p}_l$. Measurements can be points or planes, and their corresponding landmarks will store the set of features associated with the measurement. In case of point measurements, these features are keypoint descriptors, extracted using SIFT [25], while for the plane landmarks we store the plane parameters and the set of inlier points of the associated plane measurements. The set of all measurements in a frame $\mathcal{F}$ is given by $\mathcal{P} = \{\mathbf{p}_m^1, , \ldots, \mathbf{p}_m^k\}$. A *segment* is defined as a collection of measurements in a frame, $\mathcal{S} = \{\mathbf{p}_m^i, \mathbf{p}_m^j, \ldots, \mathbf{p}_m^k\}$, and the set of all segments in a frame is denoted as $\boldsymbol{\mathcal{S}} = \{\mathcal{S}^1, \ldots, \mathcal{S}^n\}$. Note that a plane can also be used to initialize a segment, as depicted in Fig. 2.

A *keyframe*, $\mathcal{KF}$, is an RGB-D frame, which is added to a map. In our system, a frame is marked as keyframe if its registered pose, in a common reference, differs sufficiently from the registered poses of all the other keyframes in a map. Our system maintains a set of maps $\boldsymbol{\mathcal{M}} = \{\mathcal{M}^1, \ldots, \mathcal{M}^n\}$, where each map is an independent collection of keyframes and landmarks, as seen in Fig. 3.

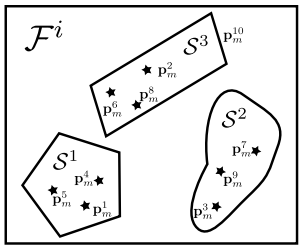

Fig. 2: Symbolic depiction of a processed frame, $\mathcal{F}^i$. In this example, there are 9 point and one plane measurements. The plane measurement, $\mathbf{p}_m^{10}$, is used to initialize a segment, $\mathcal{S}^3$. Depth-based segmentation obtained two other segments. For this frame we have $\mathcal{P} = \{\mathbf{p}_m^1, \ldots, \mathbf{p}_m^{10}\}$, and $\mathcal{S}^1 = \{\mathbf{p}_m^1, \mathbf{p}_m^4, \mathbf{p}_m^5\}$; $\mathcal{S}^2 = \{\mathbf{p}_m^3, \mathbf{p}_m^7, \mathbf{p}_m^9\}$; $\mathcal{S}^3 = \{\mathbf{p}_m^2, \mathbf{p}_m^6, \mathbf{p}_m^8\}$.

Finally, we define sets of *inlier* and *matched* point measurements for every frame $\mathcal{F}^i$ with respect to each map $\mathcal{M}^k \in \boldsymbol{\mathcal{M}}$. The inlier set $\mathcal{I}^{i,k}$ contains all point measurements that have been successfully registered to some landmark $\mathbf{p}_l \in \mathcal{M}^k$. The set of matched measurements $\mathcal{J}^{i,k}$ will contain all point measurements, which have been matched with keypoint descriptors from some landmark in the map, using the ratio test proposed in [25].

B. Registration

The goal of the registration process is to determine the rigid body transform $\hat{\mathbf{T}}^{i,k} \in SE(3)$ from the i-th frame given to the system, to the coordinate system of the k-th map in $\boldsymbol{\mathcal{M}}$, for all maps. To this end, we employ a multi-group registration scheme, consisting in sequential *frame-based* and *segment-based* registration algorithms, which aim at solving the optimization problem,

$$\hat{\mathbf{T}}^{i,k} = \underset{\mathbf{T}^{i,k}}{\operatorname{argmin}} \sum_{\mathbf{p}_m \in \mathcal{I}^{i,k}} d(\mathbf{T}^{i,k}(\mathbf{p}_m), \mathbf{p}_l), \tag{1}$$

in a RANSAC framework, where the distance operator $d(\cdot, \cdot)$ computes the distance between features. More details on the registration algorithm are found in [23], [24], [26].

C. Segment classification

In this work, we propose a novel method of segment classification based on the outcome of the multi-group registration procedure and the accumulated keypoint descriptors available in each map. This classification is the cornerstone of our map-management algorithm, which allow us to create and update object hypotheses through the construction and splitting of SLAM maps.

For the current frame $\mathcal{F}^i$ and all the maps $\mathcal{M}^k \in \boldsymbol{\mathcal{M}}$, we start by classifying the set of *registered* segments,

$$\boldsymbol{\mathcal{S}}_r^k = \left\{ \mathcal{S}^j \in \boldsymbol{\mathcal{S}} \ : \ \frac{|\mathcal{S}^j \cap \mathcal{I}^{i,k}|}{|\mathcal{S}^j|} > \delta_r, \ 0 < \delta_r \leq 1 \right\}, \tag{2}$$

which are defined as segments that have a high ratio of inlier measurements $\mathbf{p}_m \in \mathcal{I}^{i,k}$ to the total amount of measurements in the segment, given by its cardinality $|\mathcal{S}^j|$. We then partition the non-registered segments into two complementary sets: *matched* and *unmatched* segments, respectively $\boldsymbol{\mathcal{S}}_m^k$ and $\boldsymbol{\mathcal{S}}_u^k$. For every map we will thus obtain

$$\begin{aligned} &\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{S}}_r^k \cup \boldsymbol{\mathcal{S}}_m^k \cup \boldsymbol{\mathcal{S}}_u^k \\ &\boldsymbol{\mathcal{S}}_r^k \cap \boldsymbol{\mathcal{S}}_m^k = \varnothing \, ; \ \boldsymbol{\mathcal{S}}_r^k \cap \boldsymbol{\mathcal{S}}_u^k = \varnothing \, ; \ \boldsymbol{\mathcal{S}}_m^k \cap \boldsymbol{\mathcal{S}}_u^k = \varnothing. \end{aligned} \tag{3}$$

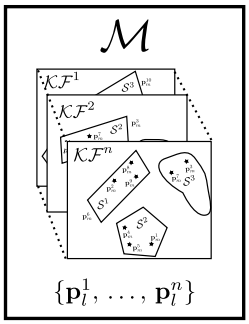

Fig. 3: A map accumulates a set of keyframes, $\mathcal{KF}^i$, and a list of landmarks, which contain the features required to match measurements in a frame with the map. In this work, we aim at building a set of independent maps, each one storing only landmarks pertaining to one object hypotheses.

A non-registered segment will belong to $\boldsymbol{\mathcal{S}}_m^k$ if we successfully match enough of its measurements' descriptors to the descriptors in $\mathcal{M}^k$,

$$\boldsymbol{\mathcal{S}}_m^k = \left\{ \mathcal{S}^j \in \boldsymbol{\mathcal{S}} \ : \ |\mathcal{S}^j \cap \mathcal{J}^{i,k}| > \alpha_m, \, \alpha_m \in \mathbb{N}^+ \right\}. \tag{4}$$

This will happen when multiple objects are associated to $\mathcal{M}^k$, and a subset of these objects is disturbed due to robotic interaction. The remaining segments are novel to the map, and are thus unregistered and unmatched, and will be assigned to $\boldsymbol{\mathcal{S}}_u^k$. Algorithm 1 illustrates this procedure, where $\mathcal{M}.registered(\mathcal{S}^j)$ and $\mathcal{M}.keypointMatched(\mathcal{S}^j)$ correspond to the inequalities in eqs. (2) and (4), respectively.

D. Detecting, tracking and reconstructing objects in the environment

The goal of our system is to detect objects in the robot environment, and to track them while building a 3D object model, through IP. This is achieved by iterating over the set of maps $\boldsymbol{\mathcal{M}}$, and updating, destroying and creating new maps based on the outcome of successive interactions with the environment. We assume that the environment will be disturbed only as a consequence of these interactions.

We start by processing every new frame provided to the system, and obtain the registration transformations from eq. (1). For every map $\mathcal{M}^k \in \boldsymbol{\mathcal{M}}$, we then execute Algorithm 1. This results in a per-map partition of the available segments, which follows eq. (3).

Once a map $\mathcal{M}^k$ has $\boldsymbol{\mathcal{S}}_m^k \neq \varnothing$, we *split* $\mathcal{M}^k$ and generate two new maps, one where we store the measurements of $\boldsymbol{\mathcal{S}}_r^k$ as landmarks, the other with the measurements from $\boldsymbol{\mathcal{S}}_m^k$. The original map is removed from the system.

This process allows our system to detect object hypotheses as the set of segments that moved with respect to the dominant motion pattern of a map, and to additionally improve on these hypotheses through subsequent map splitting. Fig. 4 depicts a simple example of map splitting.

When $\boldsymbol{\mathcal{S}}_m^k = \varnothing$, we update each $\mathcal{M}^k$ with the measurements of $\boldsymbol{\mathcal{S}}_r^k$, if the current frame is determined to be a keyframe to that map. We obtain a 3D reconstruction of any given map by recovering all the registered segments from all the keyframes in the map.

Algorithm 1 Segment Classification

Input: $\mathcal{M}$, $\boldsymbol{\mathcal{S}}$
Output: $\boldsymbol{\mathcal{S}}_r$, $\boldsymbol{\mathcal{S}}_m$ and $\boldsymbol{\mathcal{S}}_u$

procedure SEGMENTCLASSIFICATION
  $\boldsymbol{\mathcal{S}}_r \leftarrow \{\varnothing\}$
  $\boldsymbol{\mathcal{S}}_m \leftarrow \{\varnothing\}$
  $\boldsymbol{\mathcal{S}}_u \leftarrow \{\varnothing\}$
  $j \leftarrow 1$
  while $j \leq |\boldsymbol{\mathcal{S}}|$ **do**
    if $\mathcal{M}$.*registered*($\mathcal{S}^j$) is *true* **then**
      $\boldsymbol{\mathcal{S}}_r \leftarrow \boldsymbol{\mathcal{S}}_r \cup \{\mathcal{S}^j\}$
    else
      if $\mathcal{M}$.*keypointMatched*($\mathcal{S}^j$) is *true* **then**
        $\boldsymbol{\mathcal{S}}_m \leftarrow \boldsymbol{\mathcal{S}}_m \cup \{\mathcal{S}^j\}$
      else
        $\boldsymbol{\mathcal{S}}_u \leftarrow \boldsymbol{\mathcal{S}}_u \cup \{\mathcal{S}^j\}$
    $j \leftarrow j + 1$

Algorithm 2 Map Management

Input: $\overline{\boldsymbol{\mathcal{M}}}$, new RGB-D frame $\mathcal{F}$
Output: Updated $\overline{\boldsymbol{\mathcal{M}}}$

procedure MAPMANAGEMENT
  $\boldsymbol{S} \leftarrow$ *getSegments*($\mathcal{F}$)
  $\boldsymbol{\mathcal{M}}_{\text{temp}} \leftarrow \overline{\boldsymbol{\mathcal{M}}}$
  for all $\mathcal{M} \in \boldsymbol{\mathcal{M}}_{\text{temp}}$ **do**
    $\boldsymbol{\mathcal{S}}_r, \boldsymbol{\mathcal{S}}_m, \boldsymbol{\mathcal{S}}_u \leftarrow$ SEGMENTCLASSIFICATION($\mathcal{M}, \boldsymbol{\mathcal{S}}$)
    if $\boldsymbol{\mathcal{S}}_m \neq \varnothing$ **then**
      $\mathcal{M}' \leftarrow$ *newMap*($\boldsymbol{\mathcal{S}}_r$)
      $\mathcal{M}'' \leftarrow$ *newMap*($\boldsymbol{\mathcal{S}}_m$)
      $\overline{\boldsymbol{\mathcal{M}}} \leftarrow \overline{\boldsymbol{\mathcal{M}}} \setminus \mathcal{M}$
      $\overline{\boldsymbol{\mathcal{M}}} \leftarrow \overline{\boldsymbol{\mathcal{M}}} \cup \{\mathcal{M}', \mathcal{M}''\}$

E. Handling map contamination

A significant challenge of maintaining multiple independent maps lies in handling map contamination. When interacting with objects in cluttered scenes, it is common for tracked objects to get in close proximity with other tracked objects or objects in the static scene. When this happens, a new frame's segment can contain elements belonging to two or more maps, Fig. 5. We handle these cases by adopting the assumption that $\boldsymbol{\mathcal{S}}_r$ and $\boldsymbol{\mathcal{S}}_m$ cannot overlap for two different maps. We refrain from updating two maps $\mathcal{M}^k, \mathcal{M}^l \in \boldsymbol{\mathcal{M}}, k \neq l$ if any of the following conditions are satisfied: **a**) $\boldsymbol{S}_r^l \cap \boldsymbol{S}_r^k \neq \varnothing$; **b**) $\boldsymbol{S}_m^l \cap \boldsymbol{S}_m^k \neq \varnothing$, or **c**) $\boldsymbol{S}_r^l \cap \boldsymbol{S}_m^k \neq \varnothing$. In other words, a segment cannot: **a**) be registered simultaneously to two maps, **b**) be feature-matched to two maps or **c**) be registered to a map and feature-matched to another. When this happens, we assume that there is a risk of contamination between those maps. Maps in risk of contamination are still tracked, but not updated with new keyframes.

We thus create $\overline{\boldsymbol{\mathcal{M}}} \subset \boldsymbol{\mathcal{M}}$, where we keep only the maps without risk of contamination,

$$\overline{\boldsymbol{\mathcal{M}}} = \left\{ \begin{array}{c} \boldsymbol{S}_r^k \cap \boldsymbol{S}_r^l = \varnothing \wedge \\ \mathcal{M}^k \in \boldsymbol{\mathcal{M}} : \boldsymbol{S}_r^k \cap \boldsymbol{S}_m^l = \varnothing \wedge, \forall \mathcal{M}^{l \neq k} \in \boldsymbol{\mathcal{M}} \\ \boldsymbol{S}_m^k \cap \boldsymbol{S}_m^l = \varnothing \end{array} \right\}. \quad (5)$$

The map management algorithm proposed in section III-D is executed on this subset, following Algorithm 2.

F. Unregistered and unmatched segments

In some occasions, there can be one or more unregistered and unmatched segments for all maps,

$$S^j \in \boldsymbol{\mathcal{S}}_u^k, \forall \mathcal{M}^k \in \boldsymbol{\mathcal{M}}. \quad (6)$$

The circumstances that lead to this vary depending on the segmentation algorithm used to generate each frame's segments. In our work, we assume that the segmentation algorithm will tend to under-segment the observed scene. As such, segments that follow (6) will occur predominantly when a previously occluded object is revealed after an interaction, or if an object pose changes in such a way that a completely new side of it is revealed. We address these novel segments by adding them to the map with the closest registered pose to the segments' centroids, as further interactions will allow the system to correct the affected object hypothesis through map splitting.

IV. INTEGRATED SYSTEM

Section III describes a perception algorithm for detecting, tracking and modelling 3D objects that is built under the assumption that changes between RGB-D frames are due to actions exerted on the observed environment. To this end, we integrate the perception algorithm in a robotic manipulator from Fetch robotics[1], and design a simple set of manipulation primitives to act on an observed scene. A schematic depiction of the implemented system can be seen in Fig. 1.

A. Chosen segmentation algorithm

Our map management algorithm relies on the segment classification procedure illustrated in Algorithm 1. While the described approach is general, it requires segments with enough keypoint information to apply eqs. (2) and (4). Our implementation first extracts the dominant planes in the scene, such as walls and the supporting table plane, and marks these as segments. Since the employment of SIFT results in a need for reasonably large segments, to allow robust classification, we opt to under-segment the remaining points in each RGB-D frame with the Euclidean cluster extraction method from the PCL library[2]. Together with the plane segments, this strategy allows us to obtain clusters of objects on a planar surface as single segments.

B. Pushing

Similar to other works in interactive perception [4], [5], [7], [9], we rely on pushing primitives to interact with observed segments on a scene. Given a target position and a pushing direction, we position the robot end-effector behind the target and move it linearly along the pushing direction, for a pre-configured distance, with an optional small angular motion defined around the pushing target. This angular component allows the system to impart a larger rotation on the target, which is useful to gather different points of view of the object. We take into account workspace constraints by modifying the pushing distance if the resulting final position is outside the workspace boundaries.

[1] https://fetchrobotics.com/research-platforms/fetch-mobile-manipulator/
[2] http://pointclouds.org/documentation/tutorials/cluster_extraction.php

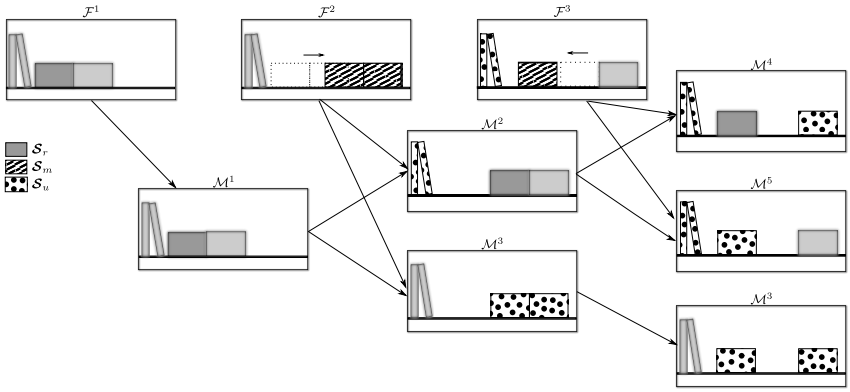

Fig. 4: Illustration of the map splitting procedure. On the top, three consecutive frames are presented to the system. A scene disturbance occurs in between each frame. The segment partition is illustrated for the frames with respect to the map that was split, i.e., segments are outlined for $\mathcal{F}^1$ and $\mathcal{F}^2$ with respect to $\mathcal{M}^1$ and $\mathcal{F}^3$ is outlined with respect to $\mathcal{M}^2$. The final set of maps is $\boldsymbol{\mathcal{M}} = \{\mathcal{M}^3, \mathcal{M}^4, \mathcal{M}^5\}$. Filled gray-scale colors indicate object which belong to $\boldsymbol{\mathcal{S}}_r$.

(a) (b) (c)

Fig. 5: Illustration of map contamination. Two objects are being modelled by two independent maps which, due to interaction, are brought in contact, Figs 5a and 5b, respectively. If the depth-based segmentation algorithm segments both objects together, the maps might be updated incorrectly, generating an incorrect model, Fig. 5c.

We select a pushing direction heuristically, by computing an artificial potential at the chosen target position, where all the detected segment centroids have a repulsive potential of $U(r_i) = 1/r_i$, where r_i is the distance from the centroid i to the target position. From this computed direction we can extract an angle α_p, with respect to a reference frame on the plane. We use this angle as the first moment α_p of a Gaussian distribution with variance σ_p^2, from which we sample the actual pushing angle, $\alpha \sim \mathcal{N}(\alpha_p, \sigma_p^2)$. This stochastic component helps the system to avoid falling in local minima, where a clump of objects is pushed back and forth without being separated. In addition to segment centroids, we consider the supporting plane limits: the limit's closest point to the target is added as a repulsive point.

C. Grasping

We leverage the reconstructed models of our object hypotheses to inform a grasp pose detector [27]. Given a target object hypothesis, we reconstruct its model as a point cloud and send it to the detector, which produces a set of proposals ranked in terms of how likely it is for the proposed pose to result in a stable grasp. We cycle from the highest to the lowest ranked grasp proposal and remove the proposals that violate workspace constraints. We then compute the inverse kinematics (IK) solutions of the remaining proposals using TRAC-IK [28]. The highest ranking proposal with a valid IK solution is chosen as the grasp pose.

The authors of [27] use a stereo depth sensor configuration to obtain a more complete partial-view point cloud of the observed scene. A remarkable advantage of modelling the objects in the environment using SLAM is the ability to accumulate different viewpoints of the tracked objects with a single static depth sensor, and using the reconstructed point cloud to inform the grasp planner. We show how this accumulation of viewpoints enables our system to obtain better grasp proposals in section V-B.

D. Interaction logic

To demonstrate our proposed method, we implement a simple interaction logic. In every iteration, we randomly sample a map from $\boldsymbol{\mathcal{M}}$, and produce a pushing direction for the centroid of its reconstructed model, as described in section IV-B. A relevant implementation detail of our system is that we keep track of which map is modelling the static scene of the robot, i.e., the set of segments that compose the dominant motion pattern of the observed scene. For every map that is not the static map, we generate grasping proposal candidates once a sufficient number of keyframes have been registered with the map, as this implies that the corresponding object hypothesis has been interacted with without the map being split, and thus there is a higher likelihood that the hypothesis is indeed a singulated object. If the grasp fails, we attempt further pushing actions, to try and generate a more complete model for the grasp pose detector.

V. Experiments

We deployed the integrated system in a human-centric environment, where several objects in contact with each other were laid on top of a table, which constitutes the supporting plane where pushing directions will be defined. The objects used in our experiments are seen from the robot viewpoint in Fig. 6. For our experiments, we laid the objects in moderately cluttered conditions, in groups of 2, 3 and 4 objects, of which

Fig. 6: The objects used in our experiments, from the robot point of view. We plan our pushing actions on the supporting plane.

Fig. 8a is an example. We then ran the interaction logic from section IV-D. The perception algorithm was executed on an Intel Core i7-7700K CPU at 4.20GHz, with 64GB of available memory. This allows the sparse SLAM method to work at a rate of between 1 and 2 Hz, and the addition of multiple map management and segment classification meant a total delay of about 2 seconds between interactions. The segment classification parameters from equations (2) and (4) were set as $\delta_r = 0.7$ and $\alpha_m = 10$. The standard deviation of the push direction distribution was set as $\delta_p = 0.4$ rad. As our segment classification system relies on rich keypoint information, we ignored segments where $|\mathcal{S}^j| < 20$. A frame $\mathcal{F}^i$ is added as a new keyframe to $\mathcal{M}^k \in \boldsymbol{\mathcal{M}}$ if it is registered with it and has either a translational or rotational components that differ, respectively, more than 5 cm or 0.087 rad from the latest registered keyframe in $\mathcal{M}^k$.

A. Detection, tracking and modelling objects

The ability of our SLAM system to model isolated objects has been demonstrated in [24]. We executed baseline experiments with single objects on the table to obtain models in ideal, non-cluttered circumstances, shown on the top row of Fig. 9. Results from an experiment with four objects are depicted in Fig. 8, where we show the initial configuration of the scene from the robot point of view, Fig. 8a, and the executed actions from an external perspective, Fig. 8b-8e. Every image is paired with an illustration of the reconstructed models from $\boldsymbol{\mathcal{M}}$. The experiment from Fig. 8 is shown in the submission video, together with experiments with different objects and initial workspace configurations. Object models reconstructed from experiments with clutter are displayed on the bottom row of Fig. 9.

B. Grasping

We tested the ability of our system to use the reconstructed object models to inform the grasp pose detector package from [27]. Once objects are singulated, we can plan a grasp using the reconstructed models. We tested the feasibility of this approach on some of the models by assuming that singulation is achieved once a map accumulates more than two keyframes, in experiments with multiple objects.

To test how adding information on existing object models helps in obtaining better grasp pose proposals, we ran experiments where we attempt a grasp after every push on an object hypothesis. While it is possible to obtain a successful grasping pose for maps with a single registered keyframe, we observed that more intricate geometries, such as the spray bottle, benefited from accumulating a higher number of keyframes, Fig. 10.

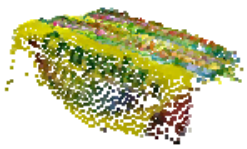

Fig. 7: Crayola box model extracted from interactive perception experiments with a single crayola box isolated on the table, using the method from [17].

VI. Discussion

We provide proof of concept qualitative results of the ability of our system to detect, track and model objects in an observed scene through IP. The performance of our approach is strongly tied to the underlying depth-based segmentation and keypoint descriptors chosen for the implementation. In our implementation, we used SIFT keypoints for detecting and representing point features, and Euclidean segmentation for extracting non-planar segments. This constrains our system to function only with significantly textured objects. Thus, we opt for under-segmenting the depth signal to ensure that detected segments will tend to have a sufficient number of keypoints. Current advances in learning keypoint descriptors [29] indicate that an extension to less textured objects can be obtained, with minimal changes to the reported pipeline.

We successfully model several objects in very challenging scenarios where the camera perspective is kept fixed and significantly far away from the objects. State-of-the-art dense methods [17] struggle to detect a single object in these circumstances, Fig. 7. The challenge of constructing a complete robotic system means that integrating alternative perception methods is a contribution in itself and as such we do not report a quantitative comparison with such methods[3].

Our heuristic choice of pushing directions is a viable way to showcase the feasibility of the perception algorithm. It is however common for the singulation of objects to not be achieved in a short number of actions. While existing IP methods [7]–[11] do not require singulation, they do not accumulate visual data over iterations, which we do through our sparse-SLAM implementation. To avoid map contamination, we are however required to singulate an object before it is modelled independently from the remaining clutter.

The major contribution of this paper is on system integration which consists of components such as scene segmentation, descriptor extraction and object pushing. While we acknowledge the challenge of end-to-end learning for the overall integrated system. The modularity of our components facilitates their replacement with equivalent learning-based methods. For example, recent work [30] explores artificial neural networks to propose pushing directions with the goal of object singulation. This would be a suitable replacement to our implementation, which could significantly increase the performance of the integrated system. Because the inability

[3]A video of our IP pipeline running with the method from [17] as a swap-in replacement for our SLAM algorithm is available upon request, and highlights these challenges.

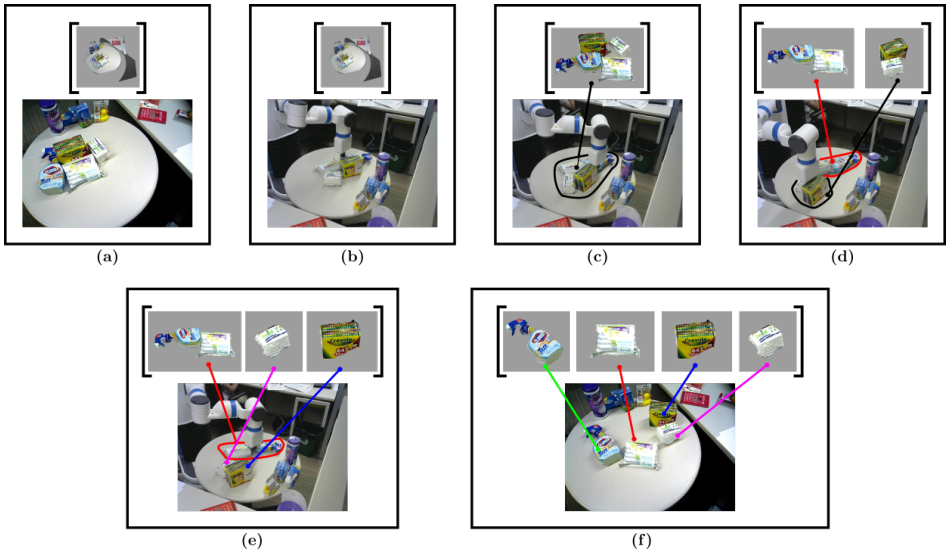

Fig. 8: The system performs a sequence of interactions to produce models of objects in its workspace. We depict the robot workspace at sequential steps in time, respectively in its initial configuration, Fig. 8a; during the four interactions with the environment, Fig. 8b-8e; and at the final configuration, Fig. 8f. At each depicted time step, we show the reconstructed models of the maps maintained by the system. For the sake of conciseness, we ommit the map of the static scene from Fig. 8c onwards. The scene images in Fig. 8a and 8f are obtained from the robot perspective. All frames given to the system come from that perspective.

Fig. 9: Top row: Object models obtained from single object scenarios with multiple interactions. Bottom row: Object models obtained from the IP experiments with clutter. Models and experiments videos available on request.

to singulate objects in a scene is the predominant cause of failure in the system, this prevents our perception algorithm to correctly model the distinct objects. In addition, our assumption of planar actions make our system unable to deal explicitly with piles of stacked objects. It is worth noting that although end-to-end learning of the overall integrated system can be feasible, it is extremely challenging considering our objective on incremental learning of the object models.

In this work, we assume that each RGB-D frame contains one single instance of each object, but place no limitation on the total number of distinct objects present in each frame. Handling multiple instances of the same object in a single frame by utilizing the estimated poses of the segments is being considered for future work. Namely, if two different segments are associated to the same map with different poses, this might be an indication of multiple instances. Reasoning geometrically to detect and track multiple object instances has been done in previous work [26], [31]. However, in the context of our system, additional care must be taken to robustly prevent map contamination in scenarios with multiple object instances.

VII. Conclusion

We detailed an algorithmic method to reconstruct 3D object models in an IP scenario, where objects are detected and tracked over consecutive robotic interactions. This is achieved through the employment of a novel segment-classification algorithm and the management of multiple, independent SLAM maps. We show results in an integrated system which has an extended scope when compared to previous IP works [3]–[11], as it not only tests segment hypotheses through interaction, but also accumulates information on existing hypotheses which enable 3D model reconstruction. Unlike [15]–[21], we reconstruct 3D models with a single, static, point of view, and maintain each hypothesis as an independent SLAM map, relying on purposeful, non-prehensile, robotic interaction to obtain new perceptual information. We illustrate how the obtained object models can be used to inform a grasp planner, and how accumulating different points of view benefits the computation of grasp pose proposals.

References

[1] Dov Katz and Oliver Brock. Interactive perception: Closing the gap between action and perception. In *ICRA 2007 Workshop: From*

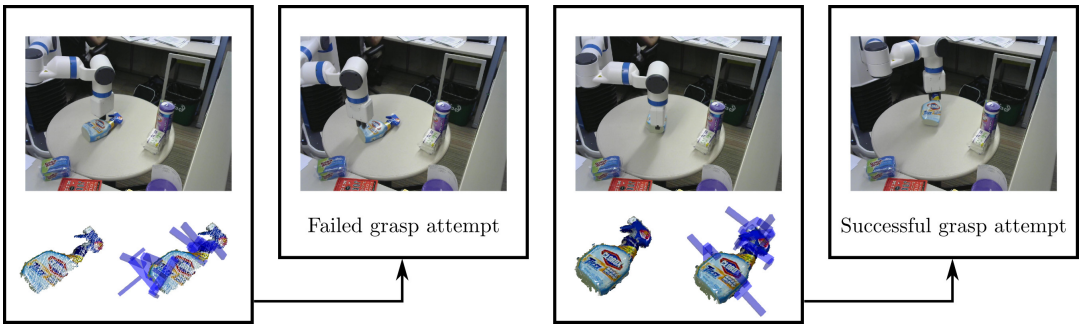

Fig. 10: Accumulating keyframes on an object hypothesis allows the system to compute better grasp proposals. In this example, the addition of more keyframes improves the detail on the bottle's neck, which leads to higher ranked grasp proposals around it, enabling a successful grasp. Grasps that lead to a collision with the table, or violate the robot kinematic constraints are filtered out. The five grasp proposals from [27] with the highest grasp quality ranking are depicted as blue parallel-jaw gripper representations.

features to actions-Unifying perspectives in computational and robot vision, 2007.

[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. Robotics*, 33(6):1273–1291, Dec 2017.

[3] M. Gupta and G. S. Sukhatme. Using manipulation primitives for brick sorting in clutter. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 3883–3889, May 2012.

[4] M. Gupta, J. Mller, and G. S. Sukhatme. Using manipulation primitives for object sorting in cluttered environments. *IEEE Transactions on Automation Science and Engineering*, 12(2):608–614, April 2015.

[5] L. Chang, J. R. Smith, and D. Fox. Interactive singulation of objects from a pile. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 3875–3882, May 2012.

[6] Niklas Bergström, Carl Henrik Ek, Mårten Björkman, and Danica Kragic. Scene understanding through autonomous interactive perception. In *Computer Vision Systems*, pages 153–162. Springer Berlin Heidelberg, 2011.

[7] Christian Bersch, Dejan Pangercic, Sarah Osentoski, Karol Hausman, Zoltan-Csaba Marton, Ryohei Ueda, Kei Okada, and Michael Beetz. Segmentation of cluttered scenes through interactive perception. In *RSS Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, pages 9–13, 2012.

[8] K. Hausman, F. Balint-Benczedi, D. Pangercic, Z. Marton, R. Ueda, K. Okada, and M. Beetz. Tracking-based interactive segmentation of textureless objects. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 1122–1129, May 2013.

[9] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz. Clearing a pile of unknown objects using interactive perception. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 154–161, May 2013.

[10] D. Schiebener, A. Ude, and T. Asfour. Physical interaction for segmentation of unknown textured and non-textured rigid objects. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 4959–4966, May 2014.

[11] H. van Hoof, O. Kroemer, and J. Peters. Probabilistic segmentation and targeted exploration of objects in cluttered environments. *IEEE Trans. Robotics*, 30(5):1198–1209, Oct 2014.

[12] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. IEEE Int'l Symp. Mixed and Augmented Reality (ISMAR)*, pages 127–136, Oct 2011.

[13] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6):169:1–169:11, November 2013.

[14] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.

[15] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard. Toward lifelong object segmentation from change detection in dense RGB-D maps. In *2013 European Conference on Mobile Robots*, pages 178–185, Sept 2013.

[16] Kai Xu, Hui Huang, Yifei Shi, Hao Li, Pinxin Long, Jianong Caichen, Wei Sun, and Baoquan Chen. Autoscanning for coupled scene reconstruction and proactive object analysis. *ACM Trans. Graph.*, 34(6):177:1–177:14, October 2015.

[17] M. Rünz and L. Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 4471–4478, May 2017.

[18] S. Choudhary, A. J. B. Trevor, H. I. Christensen, and F. Dellaert. SLAM with object discovery, modeling and mapping. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 1018–1025, Sept 2014.

[19] R. Ambrus, J. Ekekrantz, J. Folkesson, and P. Jensfelt. Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 5678–5685, Sept 2015.

[20] T. Fulhammer, R. Ambrus, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze. Autonomous learning of object models on a mobile robot. *IEEE Robotics and Automation Letters*, 2(1):26–33, Jan 2017.

[21] R. Ambrus, N. Bore, J. Folkesson, and P. Jensfelt. Autonomous meshing, texturing and recognition of object models with a mobile robot. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 5071–5078, Sept 2017.

[22] N. Bore, P. Jensfelt, and J. Folkesson. Multiple Object Detection, Tracking and Long-Term Dynamics Learning in Large 3D Maps. *ArXiv e-prints*, January 2018.

[23] Y. Taguchi, Y. Jian, S. Ramalingam, and C. Feng. Point-plane SLAM for hand-held 3D sensors. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 5182–5189, May 2013.

[24] S. Caccamo, E. Ataer-Cansizoglu, and Y. Taguchi. Joint 3D reconstruction of a static scene and moving objects. In *Proc. Int'l Conf. 3D Vision (3DV)*, pages 677–685, Oct 2017.

[25] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int'l J. Computer Vision*, 60(2):91–110, November 2004.

[26] E. Ataer-Cansizoglu and Y. Taguchi. Object detection and tracking in RGB-D SLAM via hierarchical feature grouping. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 4164–4171, Oct 2016.

[27] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt. High precision grasp pose detection in dense clutter. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 598–605, Oct 2016.

[28] Patrick Beeson and Barrett Ames. TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *Proceedings of the IEEE RAS Humanoids Conference*, Seoul, Korea, November 2015.

[29] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Proc. European Conf. Computer Vision (ECCV)*, pages 467–483, Cham, 2016. Springer International Publishing.

[30] Andreas Eitel, Nico Hauff, and Wolfram Burgard. Learning to singulate objects using a push proposal network. In *International Symposium on Robotics Research (ISSR)*, Dec 2017.

[31] W. Abbeloos, E. Ataer-Cansizoglu, S. Caccamo, Y. Taguchi, and Y. Domae. 3D object discovery and modeling using single RGB-D images containing multiple object instances. In *Proc. Int'l Conf. 3D Vision (3DV)*, pages 431–439, Oct 2017.