Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Context-Aware 3D Object Anchoring for Mobile Robots

Martin Günther[a], J.R. Ruiz-Sarmiento[b], Cipriano Galindo[b], Javier González-Jiménez[b], Joachim Hertzberg[a,c]

[a] *Robotics Innovation Center at German Research Center for Artificial Intelligence (DFKI), Osnabrück Branch, 49076 Osnabrück, Germany*
[b] *Dept. of System Eng. and Automation, Instituto de Investigación Biomédica de Málaga (IBIMA), University of Málaga, Spain*
[c] *Institute of Computer Science, Osnabrück University, Wachsbleiche 27, 49090 Osnabrück, Germany*

Abstract

A world model representing the elements in a robot's environment needs to maintain a correspondence between the objects being observed and their internal representations, which is known as the *anchoring problem*. Anchoring is a key aspect for an intelligent robot operation, since it enables high-level functions such as task planning and execution. This work presents an anchoring system that continually integrates new observations from a 3D object recognition algorithm into a probabilistic world model. Our system takes advantage of the contextual relations inherent to human-made spaces in order to improve the classification results of the baseline object recognition system. To achieve that, the system builds a graph-based world model containing the objects in the scene (both in the current and previously perceived observations), which is exploited by a *Probabilistic Graphical Model* (PGM) in order to leverage contextual information during recognition. The world model also enables the system to exploit information about objects beyond the current field of view of the robot sensors. Most importantly, this is done in an *online* fashion, overcoming both the disadvantages of single-shot recognition systems (*e.g.*, limited sensor aperture) and offline recognition systems that require prior registration of all frames of a scene (*e.g.*, dynamic scenes, unsuitability for plan-based robot control). We also propose a novel way to include the outcome of local object recognition methods in the PGM, which results in a decrease in the usually high model

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

learning complexity and an increase in the system performance. The system performance has been assessed with a dataset collected by a mobile robot from restaurant-like settings, obtaining positive results for both its data association and object recognition capabilities. The system has been successfully used in the RACE robotic architecture.

Keywords: Context-Aware Anchoring, Anchoring, World Modeling, Data Association, Mobile Robotics, Conditional Random Fields

1. Introduction

Mobile robots need to create and maintain internal representations of their surroundings for planning and executing tasks involving elements in them. Traditional tasks like navigation or localization have already well-suited solutions for building such representations, for creating metric [16, 73], topological [53, 51] or hybrid maps [72, 6]. More sophisticated *world models* arose for dealing with higher-level tasks, called *semantic maps* [37, 25, 26], which codify information from the exploration of the environment, but also consider semantic knowledge (or meta information) about the elements that can be found in the robot workspace, their properties, and their relations. Unlike the metric and topological maps case, the algorithms for building and exploiting these models are not that well-defined, and there is still significant room to explore.

One of the most critical steps during the building of these maps is creating and maintaining the correspondence between the object percepts detected in the workspace and their conceptual representation in the world model. This is known as the *anchoring problem*: "We call *anchoring* the process of creating and maintaining the correspondence between symbols and sensor data that refer to the same physical objects. The *anchoring problem* is the problem of how to perform anchoring in an artificial system." [12, p. 86f]. If we move our discourse away from semantic maps, anchoring would be still necessary for any system pursuing a plan-based robot control, where symbols are just object identifiers or labels used by the planner. However, notice the potential of semantic

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

maps where symbols are further linked to concepts codifying functionalities and relations.

The anchoring process links object percepts to their conceptually defined categories, *e.g.*: spoon, knife, mug, etc. This linking is accomplished by an object recognition method, which assigns a category to the percept. The kind of recognition algorithms traditionally exploited for this aim are referred to as *local object recognition methods* within this paper, since they work by individually classifying the perceived percepts according to their local features, *e.g.* size, shape, appearance, etc. [80]. However, it is well-known that local recognition methods are prone to provide ambiguous results [48, 13, 23], which can lead to wrong linkings in the anchoring process. This results in a incoherent world model and in failing task executions.

In this work we contribute an anchoring system with a distinctive feature: it does not simply copy the classification results from a local object recognition system, but instead uses the relations between objects (their spatial *context*) to improve those classification results. This is motivated by the fact that objects rarely occur in independent configurations at identically distributed locations. Rather, there is a coherent structure inherent to most real-world scenes. For example, a longish object in front of a monitor has a high probability of being a keyboard, whereas an object with the same local appearance next to a bread knife is more likely a cutting board.

In cases where local appearance features are not sufficient, contextual features can disambiguate object appearance in object recognition tasks [23]. Jointly considering context-based object categorization and anchoring as parts of a *context-aware anchoring system* benefits both subproblems: Anchoring receives better and more stable object classification results, while context-based object categorization can use contextual relations with anchored objects that extend beyond the sensor aperture.

As an example of this, consider Figure 1. Due to the position of the robot and the aperture of the RGB-D camera, only part of the table scene is visible in the current sensor frame, so the full context is not available. This poses a

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

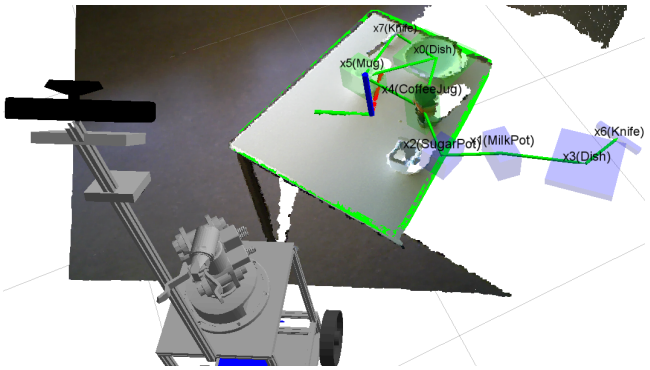


Figure 1: Robot observing a partially visible tabletop scene, along with our system's output. Notice that the RGB-D camera only captures part of the table, but the world state model maintained by the anchoring process still retains previously observed objects (blue boxes). See Figure 5 for a description of the visualization elements.

problem for most existing context-aware object recognition systems, which fall roughly into one of the following two categories. The first category is single-frame recognition systems, which recognize objects relying on single observations of the scene in the form of RGB, depth or RGB-D images [54, 76, 74, 30, 24, 64]. Regarding the exploitation of contextual information, single-frame systems are seriously limited by the sensor aperture and occlusions, given that they are able to observe only a portion of the objects and relations appearing in the whole scene. The second category are offline recognition systems, which register a number of observations prior to the recognition process in order to obtain a wider view of the scene [36, 2, 78, 75, 56, 57, 1, 41, 71]. This solves the problems caused by sensor aperture and occlusions; however, the need to finish recording the sensor data before running the object recognition process prevents online operation, which is a requirement for most plan-based robot control systems.

Our approach is to continually process single frames using a local object recognition method and integrate the object recognition results into a persistent probabilistic world model. We then use a Conditional Random Field [CRF; 35] to exploit contextual relations between objects in the current scene as well as relations with previously perceived objects from the world model to improve the

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

recognition results.

This approach has the following advantages:

- Our system can exploit contextual relations with objects that are currently out of view while still being capable of online operation (meaning that the output of the system is updated as soon as new sensor data comes in).
- The world model allows us to consistently assign the same object ID to an object without requiring that the object be constantly tracked. By anchoring symbolic object IDs to the objects reported by the local object recognition method, the planner and plan executor can refer to an object by the same symbol even after the object has disappeared from view for a prolonged period.
- The proposed system can integrate any state-of-the-art local object recognition system that processes single frames and supplement it with additional context information, achieving a significant boost in classification accuracy at a low computational overhead.

The next section relates our work to the state of the art. Section 3 describes the role of our system within the RACE project. In Section 4, we describe the proposed context-aware anchoring system, and experimental results are reported in Section 5. Finally, Section 6 contains the conclusions and possible future work.

2. State of the Art

This section starts with the discussion of related works concerning the two traditional ways to exploit contextual relations for object recognition: based on (offline) full-scene (Section 2.1), or on single-frame processing (see Section 2.2). Then, a number of relevant works addressing the anchoring problem are reported (see Section 2.4).

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

2.1. Offline, full-scene context-aware object recognition

As mentioned in the introduction, the first group of related systems performs *offline* object recognition. That is, prior to object recognition, the recording of point clouds must have finished, and all point clouds must have been registered into one global point cloud. No world model is kept and updated between classifications. In contrast, the method presented in this paper continually fuses new information into a world model, which is required for robotic applications that require online feedback, such as plan-based task execution.

One of the most influential systems in this area is the one by Koppula et al. [36]. Starting from a registered point cloud of a room (obtained from RGBD-SLAM), all planar segments in a scene are extracted. An object can consist of multiple planar segments (such as "printerFront" and "printerSide"). The system uses a probabilistic graphical model (concretely, a Markov Random Field combined with structural Support Vector Machines) to label the planar segments, exploiting local features and the geometric context between segments. An extension to the original paper shows that this model can also be used for contextual object search on a robot [2]. Given a partially recognized scene, a "hallucinated" segment is inserted into several sampled locations in the scene. Based on the contextual relations to already recognized segments, the most likely location of the target segment can be inferred. The robot then moves closer to this location and repeats the one-shot recognition process.

Xiong and Huber [78] present a similar system to semantically label a registered point cloud of the interior of a whole building that was obtained using a high-definition 3D laser scanner. Planar patches are extracted and labeled according to four object categories ("wall", "floor", "ceiling" and "clutter") using a CRF that employs both local features and context between patches. A comparison between the CRF method and a method using only local features reveals that the CRF method performs better, demonstrating that context is important in distinguishing challenging objects.

Ruiz-Sarmiento et al. [57, 56] present a method which incorporates semantic knowledge from human experts into the training process of a CRF. The seman-

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

tic knowledge is modeled in an OWL-DL ontology and describes the occurrence frequency, Gaussian distribution of geometric properties, and typical geometric context of each object category. This information is used to generate a synthetic dataset for training the CRF, which reduces the need for a large real data training set. The trained CRF is then used to label planar patches extracted from a full registered point cloud. Later, in [59], they present a survey on learning approaches for CRFs, and complete a thorough comparison and experimental analysis of their applicability in the field of scene object recognition.

Valentin et al. [75] first reconstruct a mesh of the whole scene from a series of depth images. Then, a CRF on the mesh is established, with each node representing a mesh face and edges being added between neighboring faces. As unary features, the CRF uses both various geometric features as well as appearance features from the corresponding RGB images. Pairwise features between two nodes represent the similarity in orientation and color between neighboring faces. In contrast to the previously presented approaches, no planar patches or individual objects are extracted; instead, each face of the mesh is assigned a semantic label.

In contrast to the proposed system, none of the systems above incorporates confidence values from a local object recognition method. Also, no system labels individual objects with arbitrary shapes; instead, they either work in the image domain (assigning a label to each pixel), the mesh domain (assigning a label to each mesh face) or on extracted planar patches. This is different in a group of papers which has emerged from the STRANDS project, where table-top objects are segmented.

In the first of these papers [1], a Gaussian Mixture Model (GMM) is used to model the distribution of geometric features of each object category, and another set of GMMs models each object-object relation where the objects are of different category. For each object in a scene, the objects' GMMs are used to predict the most probable category. However, context is not taken into account; instead, the object-object relations are used to compute the similarity between the current scene and trained scene types.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

In the second paper [41], an underlying 3D local object recognition method assigns a confidence value for each object category to each object. Just as in the proposed system, the task is to improve the local recognition results using spatial context. Kunze et al. [41] represent context using two different types of object-object relations: Metric Spatial Relations (MSRs) and Qualitative Spatial Relations (QSRs). Based on those, two different probabilistic inference procedures are developed: a GMM-based voting scheme for the metric relations and an approach based on Bayesian inference for the qualitative relations. The two different approaches were not combined, but instead evaluated separately. In the experiments, the metric relations slightly outperformed the qualitative relations. Both approaches showed better performance than the bottom-up perception alone, demonstrating that context is helpful for object recognition.

The third paper [71] builds on the previous work by Kunze et al. [41] and applies the MSR- and QSR-based classifiers presented there to a new dataset consisting of snapshots of several tables that have been taken over a longer time period, i.e., with variations in the number and arrangement of objects. Again, the MSRs outperform the QSRs except in cases where not enough training samples are available. However, the authors note that QSRs are better suited for knowledge transfer, either between human and robot or from robot to robot.

The spatial relations used in this work (Sec. 4.3.2) resemble the MSRs used in Kunze et al. [41] and Thippur et al. [71]; however, we do not require external information about a canonical viewpoint of a scene, but only information intrinsically available to the robot.

2.2. Single-frame context-aware object recognition

Early computer vision approaches that exploit context in object recognition are based on RGB images (see Galleguillos and Belongie [23] for an overview). Torralba et al. [74] first calculate a global descriptor (the "gist") of the image to classify the image as one type of scene (e.g., street or office) and use that as a prior for the objects in the scene without exploiting the relations between objects in the scene. Hoiem et al. [30] estimate the 3D geometry of a scene from a single

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

RGB image and use the corrected 3D position and size of objects as a prior. Galleguillos et al. [24] use spatial context between objects and demonstrate that this improves accuracy for most object categories compared to only using co-occurrence. Shotton et al. [64] exploit relations between objects (represented as regions in the image) in different configurations using a CRF based on inter-pixel statistics. Xiang et al. [76] model object co-occurrence using a CRF for the task of automated image annotation (i.e., assigning multiple labels to an image without detection or segmentation of the image). Lim et al. [44] use an Ontology and a Bayesian network to exploit context between objects and improve the output of a 2D computer vision algorithm based on SIFT and color features. Their system supports incremental learning by updating objects' co-occurrences and probabilities when new recognitions are performed. In contrast to our work, they only use spatial co-occurrence probabilities between objects in the same scene, no relations between objects.

With the introduction of the Microsoft Kinect, interest in 3D computer vision has increased dramatically. Popular tasks in RGB-D scene understanding include:

1. **Image Classification:** Each input frame contains exactly one object. The task is to classify the type of object in each image, without estimating the pose of the object [for example, 42, 67, 15].

2. **Semantic Labeling / Semantic Segmentation:** This is currently the most popular task in RGB-D scene understanding [68]. The task is to label each pixel in the RGB-D image with the object category of that pixel [for example, 66, 54, 69, 14, 47, 34].

3. **3D Object Detection and Pose Estimation:** In this task, not only the objects' category, but also their 6D pose is retrieved [11, 49, 32, 38, 39, 33, 19], which is useful for actions like pick and place. An interesting example of its application is the Amazon Picking Challenge [79, 62].

By definition, the Image Classification algorithms do not take context be-

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

tween objects into account, since there is only one object in each image. Regarding the Semantic Labeling algorithms dealing with single frames and context, most of them work by recognizing over-segmented regions from the images, and then apply a spatial consistency method such as CRF [66, 54, 47, 34]. Most relevant to this paper is the third class of algorithms (3D Object Detection and Pose Estimation). However, these algorithms usually do not take context into account (except occasionally geometric consistency), but only local object features (which is why we dubbed them *local object recognition systems* here). Our system processes output from one such system and uses context between objects (even extending beyond the current sensor frame) to improve the recognition results. We have integrated two different local object recognition systems into our anchoring framework: The ROS tabletop_object_detector [11], and the spin-image based object recognition system developed within the RACE project [49, 32]. In the experiments section (5), we only report results for the latter, since the former can only handle rotationally symmetrical objects and is based on CAD models, which makes it unsuitable for the objects present in the dataset. Our system could be combined with any of the other local object recognition systems to improve its output.

2.3. Dense 3D Semantic Mapping

The approaches discussed in the previous subsections (offline and single-frame context-aware object recognition) neither build nor maintain an internal world model. In other words, they only process the single frame (or registered set of point clouds) one at a time and do not take into account objects that are not in the current input data.

In contrast, *dense 3D semantic mapping* approaches [27, 70, 46, 45] incrementally build an internal representation by integrating each incoming RGB-D frame. Each incoming RGB-D frame is processed by a semantic image labeling method to get initial recognition guesses for each pixel/point, and this information is propagated to a point cloud based representation of the environment, where CRFs (or similar methods) are exploited for ensuring spatially consis-

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

tent labels. The single frames are registered into the point cloud using either a SLAM method (i.e., with global loop closure) or visual odometry (without loop closure). The result is a point cloud of the full scene that is densely labeled with semantic categories.

Hermans et al. [27] present a dense 3D semantic mapping system that is based on a random forest classifier for semantic labeling and visual odometry for point cloud registration. Stückler et al. [70] propose a system that combines RGB-D SLAM on multi-resolution surfel maps with a random forest classifier for semantic labeling. McCormac et al. [46] combine Convolutional Neural Networks (CNNs) for semantic labeling with a 3D surfel map based RGB-D SLAM system. The system by Ma et al. [45] also uses CNNs together with an RGB-D SLAM method, but the system also imposes multi-view consistency during CNN training by transforming the CNN feature maps into a common reference system.

There are certain similarities between the proposed system and dense 3D semantic mapping approaches: Both incrementally fuse new data with an online model, which is a requirement for plan-based task execution, because the robot can immediately react to incoming sensor data. Furthermore, both exploit context from outside the current sensor frame to improve the classification results. However, the work presented here differs from dense 3D semantic mapping approaches in two aspects: the considered object categories, and critically the underlying world model. Since dense 3D semantic mapping strives for a dense labeling of all points in the environment, the considered object categories tend to be rather coarse and focus on large-scale structures, e.g. “wall”, “table”, “floor”, and a general “objects” category (from the popular NYUDv2 dataset), whereas our system focuses on distinguishing objects that are relevant for robot manipulation (e.g. “plate”, “spoon”, “fork” etc.), so the challenges faced by both types of systems are very different. Regarding the representation, the proposed system represents individual objects and their 6DoF pose, which is a requirement for manipulation, planning and execution. Also, in a dynamic environment, an object-based representation can more easily be updated from other sources

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

(such as another robot that is moving objects, or knowledge from human-robot interaction or other data sources).

2.4. Anchoring and World Modeling

The works presented so far have mainly focused on recognizing objects without distinguishing between different instances of an object type. The following approaches are different in the sense that they keep track of previously seen objects in order to support long-term robot operation, and tackling the anchoring problem (i.e., resolving object identities).

The anchoring problem was first formally introduced by Coradeschi and Saffiotti [12]. Starting from a study and characterization of the problem, the authors lay out challenges arising from the anchoring problem and present several implemented systems that perform anchoring. This approach was later enhanced by Fichtner [21, 20], where first-order logic and the Fluent Calculus were used to formally model and represent incomplete knowledge in the anchoring problem.

Blodow et al. [8] present an anchoring system that detects objects in the environment based on a clustering algorithm (but without doing object recognition). They use Markov Logic Networks (a form of statistical relational learning) to assign identities to object perceptions. Like our system, it is passive, i.e., it builds a model of the objects in the robot's environment while the robot is performing its tasks. The RoboSherlock framework [4, 7] builds on the idea of unstructured information management to create a framework for realizing robot perception systems. Different recognition algorithms and analysis engines can be integrated as so-called annotators, and linked to knowledge bases using first-order probabilistic reasoning.

Elfring et al. [17] introduce probabilistic multiple hypothesis anchoring to create and update a world model. Their algorithm can track multiple models so that various kinds of prior knowledge can be accommodated. Several different experiments were performed both on 2D and 3D data, which showed that their anchoring system can successfully perform anchoring and even correct previous

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

object-anchor associations based on new evidence.

Blumenthal et al. [10] present a scene graph based world model that allows storing objects and other elements of the robot's environment while supporting uncertainty, tracking and sharing of data. Another such framework is EnViRe [29], which is mainly geared towards navigation and planning systems. Both of these systems focus on the storage and exchange of data, without providing an anchoring functionality.

A basic anchoring process is also performed in the paper by Ruiz-Sarmiento et al. [61], where the system checks the location of two percepts, the overlapping of their bounding boxes, and their appearance to decide if they refer to the same physical element. The output of this anchoring is used to maintain a probabilistic representation of the elements in the robot workspace.

3. Anchoring in the RACE Architecture

The anchoring system presented here has been successfully employed in the context of the RACE project [28]. In RACE, all high-level modules communicate via the so-called *blackboard*, which is implemented on top of an RDF database. The elements stored on the blackboard are called *fluents*, i.e., temporally valid ground facts of a Description Logic (DL) ontology. Fluents have both a start and finish time between which they are valid. Since the main objective of the RACE project was enabling a robot to learn from its experiences, fluents have to remain available for later analysis even after no longer being valid, so a fluent is never deleted, only *ended* by setting its end time to the current time (see Listing 1 on page 42 for an example of the fluent format). Further details on the RACE blackboard can be found in [28, 55].

Figure 2 shows the role of the anchoring module presented here within the RACE architecture. All knowledge about physical objects is exchanged between modules via object fluents on the blackboard. Some of these fluents are created by external knowledge sources (for example, a module that initializes the blackboard with the locations of known objects, or a module that gathers knowledge

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

about object locations from human-robot interaction). They are read by the planner, which uses this information to create the initial state of the planning problem whenever a new plan is requested. They are both read, updated and created by the execution monitor (for example, when the robot picks up a mug from the table, the execution monitor ends the fluent that specifies that the mug is on the table and adds one that specifies that it is now being held by the robot gripper). The anchoring module's role is to anchor the object fluents on the blackboard to percepts of individual objects produced by the object recognition system and update the blackboard accordingly. This involves updating fluents (when an object has moved or a previously untracked anchor has been reacquired), ending fluents (when an object is found to no longer be in or near the location stored on the blackboard), and creating new fluents (when new objects not yet represented on the blackboard are detected).

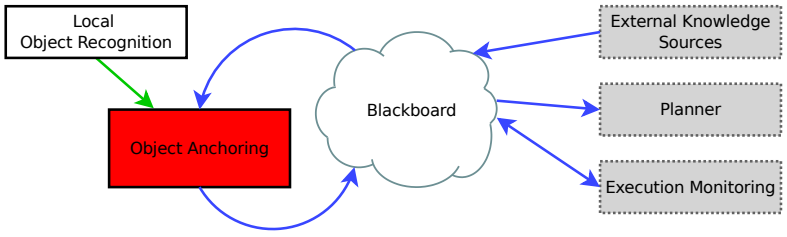


Figure 2: Overview of the role of the context-aware anchoring system within the RACE architecture. Green arrow: object recognition results; blue arrows: fluents related to physical objects.

4. Context-Aware Anchoring

The proposed context-aware anchoring system is a combination of: i) a local object recognition method, ii) an anchoring process, and iii) a Conditional Random Field (see Figure 3). The *local object recognition method*[1] (Sec. 4.1)

[1]In the following, we use the term *local* object recognition for this process to distinguish it from the *joint* object recognition performed by the CRF, which performs joint classification

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

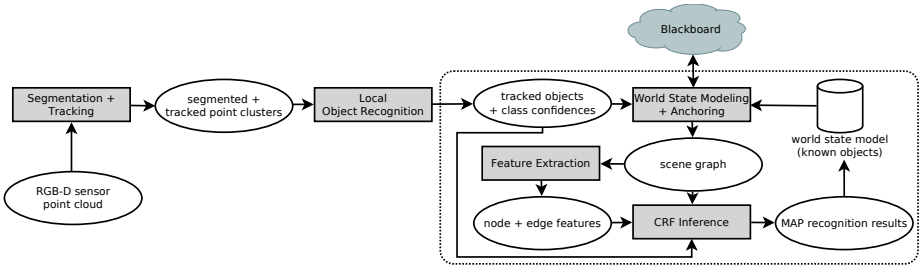


Figure 3: Overview of the proposed system. Ovals represent consumed/produced data, while boxes stand for processes. Locally tracked objects are anchored to previously observed (known) objects, and a scene graph of objects is built that includes all currently tracked objects and nearby known objects that are out of view of the camera. All objects in the scene graph are jointly classified by an inference process over a CRF, and the set of known objects is updated. Our contribution is highlighted in the dotted box.

segments the sensor point cloud into object point clusters and yields a local recognition result for each isolated object (in the form of confidence values for each object category). This provides the input for the *anchoring process* (Sec. 4.2), which updates its persistent world state model (a set of so-called *anchors*) with the new information. The anchoring module then creates a scene graph consisting of all the currently observed objects and all nearby objects that are currently out of view of the camera; it also adds edges between objects that are close to each other. A CRF is then built incorporating the nodes and edges of the scene graph along with their features, and the confidence values from the local object recognition (Sec. 4.3.2). Finally, a probabilistic inference process over the CRF computes the joint object recognition results, and the anchors are updated accordingly (Sec. 4.3).

4.1. Segmentation, Tracking and Local Object Recognition

The first steps in our processing pipeline segment the input point cloud into individual objects and perform local object recognition, which classifies each segmented object separately. The output of the local object recognition system

of all objects in the scene and takes contextual relations into account.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

is a set $X = \{x_1, \ldots, x_n\}$ of objects found in the input point cloud, and for each object its position, 3D bounding box and a vector $\boldsymbol{c}_{x_i} = [c_1, \ldots, c_k]$ of confidence values, representing the confidence with which object x_i belongs to the respective class in the set of possible categories $L = \{l_1, \ldots, l_k\}$. The objects detected by this stage are the basic units for all later stages of the pipeline.

Here, we use the spin-image based object recognition system developed within the RACE project [49, 32]. This object recognition system first performs table-top segmentation on the input point cloud: a parametric plane model is fitted to the input point cloud using random sample consensus (RANSAC), and points that lie within a threshold of the identified plane are removed. The remaining points are clustered into objects, and tiny clusters are discarded to remove outliers. Object recognition is performed on each object x_i by computing spin-image descriptors on certain keypoints of each object and comparing the result with a trained database of objects, which results in the confidence vector $\boldsymbol{c}_{x_i}$.

Apart from segmentation and local object recognition, the local object recognition system used here also provides real-time tracking of the recognized objects: a unique *track ID* is attached to all observations of the object in subsequent camera frames. Tracking is lost when the object is no longer in view of the RGB-D camera.

Note that the proposed anchoring system makes no assumptions about the local object recognition system. For instance, although the spin-based object recognition system requires that objects are separated by a certain minimum distance in order to segment them, this is not a requirement of the anchoring system. This means that the anchoring system can integrate any other 3D local object recognition method that is capable of providing a bounding box and confidence vector for each object (such as [38, 39, 33, 19]). Specifically, although we make use of the tracking functionality in Sec. 4.2, our system does not strictly require it; when tracking is not provided by the local object recognition system, our system relies on its anchoring functionality alone. In fact, we have used our system to process output from the ROS tabletop_object_detector recognition

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

system [11], which does not perform tracking, but only recognition. We did not use it in the experiments (Section 5), since it requires CAD models of all objects and can only handle rotationally symmetrical objects, which makes it unsuitable for the objects present in the dataset.

4.2. Anchoring

The anchoring module has two closely related functions:

1. creating and maintaining over time a persistent, probabilistic world model of the locations, identities and categories of objects perceived so far; and
2. grounding object names from a symbolic knowledge base to object identities in this world model.

As pointed out in the introduction, even without symbolic names for objects, having a persistent world model of objects outside the robot's field of view (1.) is already beneficial to context-based joint object classification. Grounding a shared symbolic knowledge base to this model (2.) is essential for the creation and execution of robot plans in a plan-based robot control architecture.

Our anchoring module has a persistent internal representation (the *world model*) of the objects in the environment. Following the terminology of Coradeschi and Saffiotti [12], we call the representation of an object *anchor*. An anchor in our system is a richer representation than the purely symbolic object fluents and contains:

- the symbolic name of the object on the blackboard that corresponds to this object;
- the last known 6DoF object pose (position and orientation);
- the last observed 3D bounding box;
- the vector $\boldsymbol{c}_{x_i} = [c_1, \ldots, c_k]$ of confidence values reported last by the local object recognition for this object (see previous section);
- the MAP object type computed by the CRF (see Section 4.3); and

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

- the track ID, if the object is currently tracked by the local object recognition.

Note that an anchor contains a probabilistic representation of the possible object types in the form of the vector of confidence values $\boldsymbol{c}_{x_i}$. This vector is used by the CRF (see Section 4.3) when computing context relations with anchored objects outside the current camera frustum.

An anchor in our framework can be in one of two different states (Figure 4): *untracked* or *tracked*. If an object is currently in the robot's field of view and thus being tracked by the local object recognition, we call the corresponding anchor a *tracked anchor* and otherwise an *untracked anchor*. Figure 4 also shows the functionalities that create or destroy anchors and perform the transitions between states: *Find*, *Reacquire*, *Track*, *Lose*, *Discover* and *Destroy*. The first three functionalities are inspired by Coradeschi and Saffiotti [12]; the last three are additional functionalities made necessary by the bottom-up part of our framework.

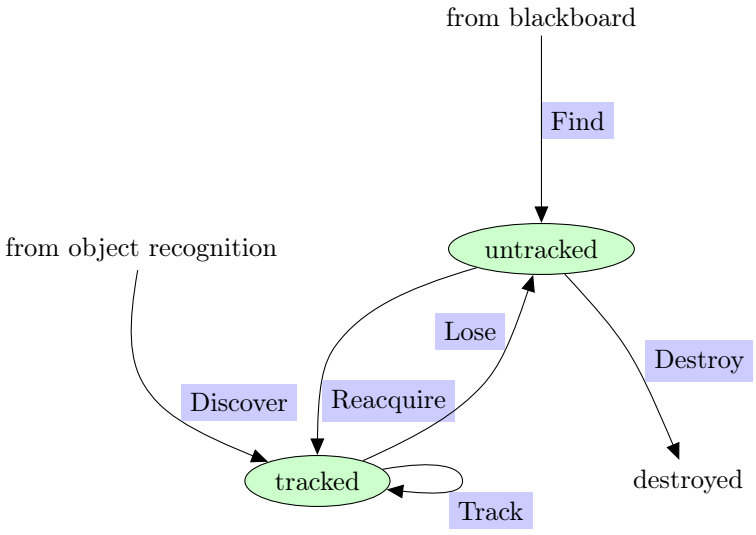


Figure 4: State diagram showing the two possible states of an anchor (green) and the functionalities that perform the transitions between them (blue).

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Find creates a new anchor whenever a new object fluent is added to the blackboard by an external module. For example, the initial knowledge loader adds initially known object fluents during startup, and the plan executor adds an object fluent when an object is placed on a table by the robot. These anchors start in the *untracked* state. As soon as a matching object is detected by local object recognition, *Reacquire* binds the anchor to this object, and the anchor enters the *tracked* state. While *Find* performs top-down anchoring, *Discover* performs bottom-up anchoring: Whenever an object is reported by the object recognition module that cannot be associated to a known untracked anchor, *Discover* creates a corresponding tracked anchor. While an object is tracked by local object recognition, *Track* updates the properties of its anchor. When it stops being tracked, *Lose* transitions the anchor to the *untracked* state. Finally, *Destroy* checks whether an object corresponding to an untracked anchor has been physically removed and deletes the corresponding anchor.

In our framework, these functionalities are implemented by executing the following steps whenever a new set of objects is reported by the local object recognition. These will be detailed in the remainder of this subsection and the following subsections.

1. **Object-Anchor Similarity Computation (Section 4.2.1):** Anchoring attempts to match the new objects to the nearby untracked anchors in the current world model. In order to do so, the *similarity* between each pair of objects is calculated.

2. **Object-Anchor Data Association (Section 4.2.2):** Using these similarity values, the best *assignment* of new objects to untracked anchors is calculated using the Hungarian method [40]. If the cost of assigning an object does not exceed a certain threshold (i.e., the object is "similar enough" to the anchor), the object is assigned to the anchor (*Reacquire*); otherwise, a new anchor is created (*Discover*). If the object is already tracked, its track ID will be used to instantly match the new observation to its anchor and update the anchor (*Track*). All tracked anchors that

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

have not been assigned a tracked object are transitioned to the untracked state (*Lose*). It is also possible to integrate a local object recognition system that does not provide tracking information. In that case, all anchors are transitioned to the untracked state and must be reacquired in the next step.

3. **Removed Object Detection (Section 4.2.3):** Check for physically *removed objects*: All remaining nearby untracked anchors are checked whether their last known bounding box is fully within the robot's current field of view and empty; if so, the anchor is removed (*Destroy*).

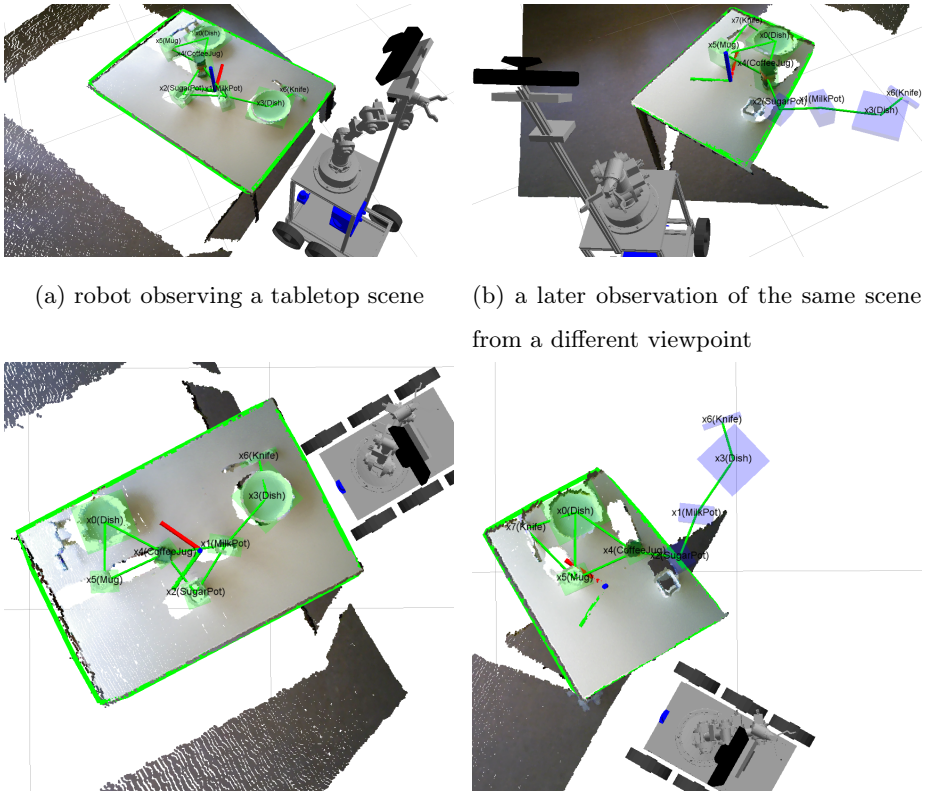
4. **Joint Object Recognition (Section 4.3):** A scene graph for the local scene is created from the tracked anchors and nearby untracked anchors (i.e, anchors of objects that are close to currently observed objects, but occluded or outside the camera frustum). Unary features (such as size or elongation) and pairwise features (such as distance between two objects or relation between two objects with respect to the table) are extracted and added to the graph. A CRF inference procedure over the graph computes the MAP object types, which are used to update the corresponding anchors.

5. **Blackboard Synchronization (Section 4.4):** The changed anchors are propagated to the corresponding object fluents in the blackboard. Also, new anchors are created for object fluents that have been added to the blackboard by an external module (*Find*).

4.2.1. Object-Anchor Similarity Computation

A prerequisite for matching newly tracked objects to existing untracked anchors is to compute the *similarity* between each tracked object and each nearby untracked anchor. Here, a higher similarity between a tracked object and an anchor means a higher likelihood that the tracked object corresponds to the physical object represented by the anchor, and thus the anchor should be assigned to that object (*reacquired*).

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016



(a) robot observing a tabletop scene

(b) a later observation of the same scene from a different viewpoint

(c) top-down view of (a)

(d) top-down view of (b)

Figure 5: Two subsequent robot observations of a tabletop scene. Transparent boxes: bounding boxes of anchored objects with their CRF classification result (CRF nodes; green: currently tracked, blue: not currently tracked); lines: context edges between nearby objects (CRF edges). Also shown is the bounding polygon of the currently observed table surface and its center.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

In our framework, there are two main features of objects that influence similarity: the difference of the two objects' classification results (specifically, the difference of their category confidence vectors $\boldsymbol{c}_{x_i}, \boldsymbol{c}_{x_i}$) and the euclidean distance between the two objects. In other words, if local object recognition reports an object with roughly the same classification in roughly the same location as one of the untracked anchors, chances are good that it is the object represented by the anchor. The importance of those two features can be explained by the assumptions we make about the information available to our algorithm and by our formulation of the problem. We assume that the set of possible categories L is already as fine-grained as possible; i.e., it is impossible to subdivide any of the categories into meaningful sub-categories based on the appearance of objects. In particular, we do not assume that any object (like "Joachim's mug") can be directly recognized based on its appearance alone. If there was such a one-to-one mapping between object appearance and object identity, the anchoring problem would be much simplified. The fact that we do not make this assumption has two consequences: first, if an object is replaced by one of similar appearance while the robot is not observing the scene, our algorithm will anchor it incorrectly; and second, if an object is moved over a large distance while the robot is not observing the scene, the anchor will be destroyed when the robot revisits the scene (see below), and the object at its new location will be assigned a new anchor when observed by the robot.

In order to compare different possible assignments of tracked objects to anchors (see next subsection), we need to combine the features (difference in classification results and distance) into a single similarity function. In order to approximate this function, a Support Vector Machine (SVM) was trained on a dataset of object/anchor pairs manually labeled as "same object"/"different object" (i.e., whether the tracked object is the same represented by the anchor).

Apart from the two features presented so far (difference of confidence vectors and distance between objects), additional SVMs were trained on different sets of features; however, the evaluation showed that the two features presented above were sufficient. All evaluated features are listed in Tab. 1. For details on

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Table 1: Similarity features for objects x_1 and x_2 with centroids $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and confidence vectors $\boldsymbol{c}_{x_1}$ and $\boldsymbol{c}_{x_2}$ (see Sec. 4.3.3). The functions `area_h`, `area_v`, `elong_h` and `elong_v` refer to the CRF unary features in Table 2 on page 37, discussed in Sec. 4.3.2.

Feature name	**Definition**	**Description**
`distance`	$\lvert\boldsymbol{p}_1 - \boldsymbol{p}_2\rvert$	Distance between object centroids
`diff_object_types (norm)`	$\lvert\boldsymbol{c}_{x_1} - \boldsymbol{c}_{x_2}\rvert$	Aggregated difference of the two objects' classification results (as reported by the local object recognition)
`diff_object_types (vector)`	$(\boldsymbol{c}_{x_1 i} - \boldsymbol{c}_{x_2 i})^2$	Component-wise difference of the two objects' classification results (one feature for each object category i)
`diff_h_area`	$(\texttt{area\_h}(x_1) - \texttt{area\_h}(x_2))^2$	Squared difference between horizontal bounding box areas
`diff_h_elong`	$(\texttt{elong\_h}(x_1) - \texttt{elong\_h}(x_2))^2$	Squared difference between horizontal bounding box elongations
`diff_v_area`	$(\texttt{area\_v}(x_1) - \texttt{area\_v}(x_2))^2$	Squared difference between vertical bounding box areas
`diff_v_elong`	$(\texttt{elong\_v}(x_1) - \texttt{elong\_v}(x_2))^2$	Squared difference between vertical bounding box elongations
`volume_ratio`	$\texttt{volume}(x_1)/\texttt{volume}(x_2)$	Ratio of bounding box volumes

the training procedure and a comparative evaluation of different sets of these features, see Section 5.3.

Using the trained SVM, the *similarity* between two objects $s(x_1, x_2)$ can easily be derived by computing the signed distance from the point in feature space that represents the object pair to the hyperplane that forms the SVM's decision boundary (in Hesse normal form). If this signed distance is zero, the object pair lies exactly on the decision boundary. Object pairs with a negative distance lie between the origin and the hyperplane and would be classified as "same object" by the SVM (and vice versa for positive distances). In general, the higher the signed distance, the less similar the two objects are to each

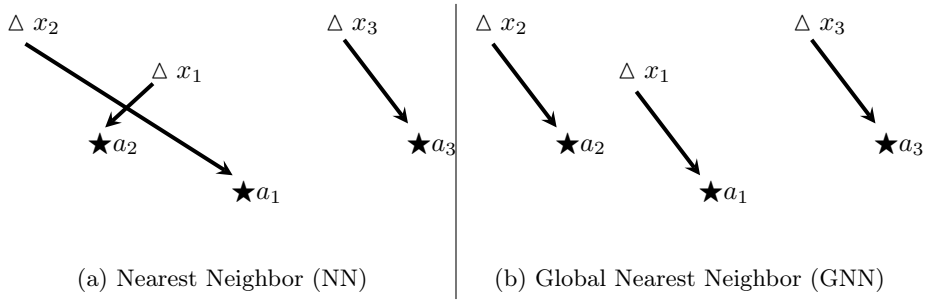
Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

other. Figure 11 on page 49 shows the decision boundary and the two classes (same/different) of object pairs. Note that the similarity function can also be used to compute the similarity between an object and an anchor $s(x, a)$, since all relevant properties are stored with each anchor.

4.2.2. Object-Anchor Data Association

Data association is "the process of determining the origin of sensor measurements" [43], i.e., the process of associating uncertain sensor measurements of objects (or environment features) to existing object tracks (or features in a map). In the context of our work, the uncertain measurements are the object detections reported by the local object recognition method which we want to associate to the existing anchors.

There are numerous approaches to the data association problem. The simplest algorithm is Nearest Neighbor (NN): each detected object is greedily associated to the nearest anchor (or rather, the anchor with the highest similarity s, as defined above). However, NN often results in a non-optimal assignment, as illustrated in Fig. 6. This problem is solved in the Global Nearest Neighbor (GNN) algorithm, which computes a jointly optimal assignment. A variant of GNN was used in our system.



(a) Nearest Neighbor (NN) (b) Global Nearest Neighbor (GNN)

Figure 6: A comparison between the assignments produced by the NN and GNN algorithms. For NN, the order of assignments (here: x_1, x_2, x_3) is significant. For simplicity's sake, this figure only shows the distance between the objects, whereas the actual assignment costs are computed from the similarity function $s(x_i, a_j)$ between object and anchor, which is based on distance and appearance. (★: anchors, △: detected objects)

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Many more elaborate alternatives to the GNN algorithm can be found in the multi-target tracking and data association literature [3]. The most prominent representatives are the Joint Probabilistic Data Association Filter [JPDAF; 9] and approximations of Multiple Hypothesis Tracking [MHT; 52]. These algorithms delay the decision which object to assign to which anchor and use information from future frames to help disambiguate the assignment. This improves the assignment accuracy especially in situations with a lot of clutter, where GNN suffers from its eager assignment strategy and can produce many false positives. However, this improved ability to deal with clutter comes at the cost of a hugely increased computation time and at a slight loss of real-time capability, since there is a fixed delay until final assignments are made. Since the local object detection method used here already performs some short-term tracking and clutter removal, GNN was deemed sufficient, which is confirmed by our experiments (Sec. 5.3).

Before the ObjectAnchorAssociation procedure is run, the set of objects reported by local object recognition is preprocessed. If an object is already tracked and associated to an anchor, its track ID will be used to instantly match the new observation to its anchor and update the anchor. Conversely, all anchors that are tracking an object which is no longer reported by the local object recognition method are transitioned to the "untracked" state. The remaining set of newly detected objects X is passed on to the ObjectAnchorAssociation procedure shown in Algorithm 1, along with a set of nearby untracked anchors A, i.e., anchors that are in some specified gating region around the objects in X and that are not associated to a currently tracked object. The output of the algorithm is a set M of assignments that assigns each object $x_i \in X$ to an anchor and an updated set of anchors $A' \supseteq A$ that includes any newly created anchors in case new objects have been discovered. The first step is to build a cost matrix $\boldsymbol{C}$, where the i-th row and j-th column represents the cost of assigning x_i to $a_j = s(x_i, a_j)$. Using these similarity values, the optimal assignment of new objects to untracked anchors is calculated using the Hungarian method [40]. Also known as the Kuhn–Munkres algorithm or Munkres assign-

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

ment algorithm, the Hungarian method solves the assignment problem in $O(n^4)$.
595 A later modification by Edmonds and Karp reduced this runtime to $O(n^3)$. The implementation used here [50] includes this optimization; it also does not require $\boldsymbol{C}$ to be a square matrix, in contrast to the original algorithm. The output of the Hungarian algorithm is a set I of index pairs of $\boldsymbol{C}$ representing the optimal assignment.

Algorithm 1 ObjectAnchorAssociation

Input: a set of newly detected objects $X = \{x_1, \ldots, x_n\}$; a set of nearby untracked anchors $A = \{a_1, \ldots, a_m\}$; a similarity function $s(x, a)$

Output: a set of object-anchor assignments $M = \{\langle x_1, a' \rangle, \ldots, \langle x_n, a'' \rangle\}$; an updated set of anchors $A' = \{a_1, \ldots, a_k\}$, where $k \geq m$

1: **function** ObjectAnchorAssociation(X, A)
2: $M \leftarrow \varnothing$
3: $A' \leftarrow A$
4: $\boldsymbol{C} \leftarrow \begin{pmatrix} s(x_1, a_1) & s(x_1, a_2) & \ldots & s(x_1, a_m) \\ s(x_2, a_1) & s(x_2, a_2) & \ldots & s(x_2, a_m) \\ \vdots & \vdots & \ddots & \vdots \\ s(x_n, a_1) & s(x_n, a_2) & \ldots & s(x_n, a_m) \end{pmatrix}$
5: $I \leftarrow$ HungarianAlgorithm($\boldsymbol{C}$)
6: **for each** $\langle i, j \rangle \in I$ **do**
7: **if** $s(x_i, a_j) < 0$ **then**
8: $M \leftarrow M \cup \{\langle x_i, a_j \rangle\}$ ▷ Reacquired anchor
9: **else**
10: $a^* \leftarrow$ CreateAnchor(x_i) ▷ Discovered new object
11: $A' \leftarrow A' \cup \{a^*\}$
12: $M \leftarrow M \cup \{\langle x_i, a^* \rangle\}$
13: **end if**
14: **end for**
15: **return** M, A'
16: **end function**

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

If the cost of assigning x_i to a_j (where $\langle i, j \rangle \in I$) does not exceed a certain threshold (i.e., the object is "similar enough" to the anchor), the object is assigned to the anchor. The natural choice for this threshold is 0 due to the definition of $s(x_i, a_j)$, where a negative value means that the SVM would classify this pair as "same object", a positive value would mean "different object", and 0 is the decision boundary.

If the cost exceeds this threshold 0, there is no anchor corresponding to this object (i.e., a new object has been discovered). In this case, a new anchor is created and the object is assigned to that anchor.

4.2.3. Removed Object Detection

Using the algorithm in the previous subsection, our system can discover new objects and reacquire and track existing ones. However, when the robot revisits a scene, it also has to check whether anchored objects have been physically removed in the meantime. Since the local object recognition method only yields positive information (a set of objects that have been detected), the algorithm described here has to process the raw input point cloud to perform this task.

The general idea behind the RemovedObjectDetection algorithm is the following. For each nearby *untracked* anchor (i.e., an anchor that could not be assigned to an existing object in the scene by the Object-Anchor Data Association), the following steps are performed:

1. check whether the bounding box of an anchor lies within the view frustum of the sensor at the current camera pose (i.e., the object is expected to be visible);

2. if yes, check whether there are any points visible inside the anchor's bounding box, or between the bounding box and the camera (that could occlude the object);

3. if the number of such points is below a certain threshold, the object is neither present nor occluded, and the anchor is destroyed.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Algorithm 2 on page 31 shows how these steps can be performed efficiently, given two requirements: The input point cloud C must be *organized* and *projectable*. An organized point cloud is a set of points that is indexed in two dimensions, similar to a matrix or organized image. A point cloud is projectable if it is organized and the 3D coordinates of all its points can be computed from the (u, v) index of a point according to a pinhole camera model. Organized, projectable point clouds are commonly produced by projective devices such as stereo cameras, Time-of-Flight cameras or structured light cameras such as the ASUS Xtion used in the experiments.

From the anchor's bounding box dimensions $\boldsymbol{d}$, the set B of the eight bounding box corner points is computed and represented in homogeneous coordinates (as a four-dimensional vector, where the last component is 1). These corner points are first transformed into the camera frame by multiplying each point with the 4×4 transformation matrix $\boldsymbol{R}$ (resulting in B') and then projected into the 2D image plane (resulting in B'' by multiplying with the 3×4 camera projection matrix $\boldsymbol{P}$. The matrix $\boldsymbol{P}$ has the form

$$\boldsymbol{P} = \begin{pmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{1}$$

where f is the camera's focal length and c_x, c_y are the coordinates of the camera's principal point. The resulting elements of B'' have the form $\boldsymbol{b''} = (u \;\; v \;\; w)^\mathsf{T}$, where w is an arbitrary scale factor, so the true pixel coordinates in the image plane are u/w and v/w. Using B' and B'', the algorithm checks whether all eight bounding box corners are inside the camera frustum. This is the case when each corner's z coordinate lies within the minimum and maximum camera measurement depth $\hat{z}, \check{z}$, and the pixel coordinates $u/w, v/w$ are between 0 and the image width $\check{u}$ and height $\check{v}$ respectively. If at least part of the bounding box lies outside the camera frustum, the algorithm can not confirm that the object was removed and thus returns false.

Switching back to 3D coordinates, the next step is to compute the convex

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

hull H of the bounding box corners in B' and the camera origin; since the elements of B' are in the camera frame, the camera origin is always at $(0\ \ 0\ \ 0\ \ 1)^\mathsf{T}$. The function ConvexHull can be implemented efficiently using Delaunay triangulation; an example result is shown in Figure 7. Delaunay triangulation splits each rectangular face of the bounding box into two triangles. To increase efficiency when checking inclusion of a point in the convex hull, such coplanar faces are removed in a post-processing step.

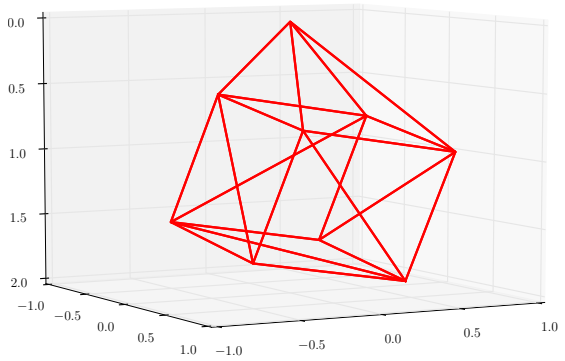


Figure 7: The convex hull of the eight corners of an object's bounding box and the camera origin (top center, at $(0\ \ 0\ \ 0\ \ 1)^\mathsf{T}$). The convex hull was calculated using Delaunay triangulation, which split the rectangular faces of the bounding box into two triangles each. Such coplanar faces are removed in a postprocessing step.

Any point inside this convex hull is either inside the object's bounding box (and therefore considered a potential point on the object), or between the camera and the bounding box (and therefore considered an occluding point). The final step of the algorithm is to count the number points inside the convex hull.

This process can be sped up by making use of the fact that the input point cloud is organized. First, we compute the image-axis-aligned 2D bounding box $(u_{min}, u_{max}, v_{min}, v_{max})$ from the minimum (resp. maximum) of the (u, v) coordinates of the eight bounding box corners in B''. Any point outside this 2D bounding box cannot be inside the 3D convex hull. Therefore, it suffices to only check the points inside the 2D bounding box for inclusion in the convex hull, greatly speeding up computation.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

The actual IsInside check (whether a point is inside the convex hull or not) is both straightforward and efficient. By representing the faces of the convex hull as planes in Hesse normal form, it only takes one vector multiplication for each of the 9 faces to check if the point lies on the "inside" side of each plane.

If the number of points inside the hull exceeds a threshold, the algorithm returns "false" (i.e., the object is either present but could not be detected by the local object recognition, or it is occluded). Otherwise, it returns "true" (the object was removed). As was discussed in Section 4.2.1, an object that has been removed while the robot was not observing the scene cannot be anchored again. For this reason, when RemovedObjectDetection returns "true" for an anchor, the anchor is destroyed.

4.3. Joint Object Recognition

As part of our context-based anchoring system, we need to solve the *joint object recognition* problem, where the aim is to assign to each of the objects in a scene their respective category (e.g., mug, dish, spoon), taking into account both the local features of each object and their contextual relations. To this end, we build a scene graph from the tracked anchors and all nearby untracked anchors, and use a Conditional Random Field [CRF; 35] to compute the most probable categories of all objects, considering their context (see Figure 5).

4.3.1. CRF Formulation for Joint Object Recognition

This joint object recognition problem is modeled as a CRF as follows. Let $X = \{x_1, \ldots, x_n\}$ be a set representing n observed objects within a given scene, where each object x_i is characterized through a vector of m features $\boldsymbol{f}_{x_i u} = (f_{x_i u_1} \ \ldots \ f_{x_i u_m})^\mathsf{T}$, e.g., size, height or elongation, $L = \{l_1, \ldots, l_k\}$ the set of the k possible object categories, and $Y = \{y_i, \ldots, y_n\}$ a set of discrete random variables over L, that assigns to each object in X a category from L. Thus, the joint object recognition problem in the CRF framework consists of maximizing the probability distribution $P(Y|X)$, i.e., to find the most probable categories assignment to Y given the characterized objects' observations in X.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Algorithm 2 RemovedObjectDetection

Input: an untracked anchor's bounding box dimensions $\boldsymbol{d} = (d_x \;\; d_y \;\; d_z)^\mathsf{T}$ and pose, given as a rigid transformation $\boldsymbol{R}$ from the bounding box frame to the camera frame; an organized, projectable point cloud $C = \{\boldsymbol{p}_{11}, \boldsymbol{p}_{12}, \ldots, \boldsymbol{p}_{\check{u}\check{v}}\}$, where $\check{u}$ is the width and $\check{v}$ is the height of the image; a camera projection matrix $\boldsymbol{P}$; minimum and maximum camera measurement depth $\hat{z}, \check{z}$; a threshold on the number of points n below which the object is considered removed

Output: true if the object was removed, false if occluded or outside camera field of view

function RemovedObjectDetection($\boldsymbol{d}, \boldsymbol{R}, C, \check{u}, \check{v}, \boldsymbol{P}, \hat{z}, \check{z}, n$)

$B \leftarrow \left\{ \begin{pmatrix} -d_x/2 \\ -d_y/2 \\ -d_z/2 \\ 1 \end{pmatrix}, \begin{pmatrix} -d_x/2 \\ -d_y/2 \\ d_z/2 \\ 1 \end{pmatrix}, \ldots, \begin{pmatrix} d_x/2 \\ d_y/2 \\ d_z/2 \\ 1 \end{pmatrix} \right\}$

$B' \leftarrow \{\boldsymbol{R}\boldsymbol{b} \mid \boldsymbol{b} \in B\}$

$B'' \leftarrow \{\boldsymbol{P}\boldsymbol{b'} \mid \boldsymbol{b'} \in B'\}$

if $\exists \boldsymbol{b'} = (x \;\; y \;\; z \;\; 1)^\mathsf{T} \in B' \colon z \notin [\hat{z}, \check{z}]$ **then**

  return false

else if $\exists \boldsymbol{b''} = (u \;\; v \;\; w)^\mathsf{T} \in B'' \colon (u/w \notin [0, \check{u}]) \vee (v/w \notin [0, \check{v}])$ **then**

  return false

else

  $H \leftarrow \text{ConvexHull}\left(B' \cup \left\{(0 \;\; 0 \;\; 0 \;\; 1)^\mathsf{T}\right\}\right)$

  $u_{min}, u_{max}, v_{min}, v_{max} \leftarrow \min_{u/w}\left\{(u \;\; v \;\; w)^\mathsf{T} \in B''\right\}, \ldots$

  $C' \leftarrow \{\boldsymbol{p}_{uv} \in C \mid (u_{min} \leq u \leq u_{max}) \wedge (v_{min} \leq v \leq v_{max}) \wedge \text{IsInside}(\boldsymbol{p}_{uv}, H)\}$

  return $|C'| \geq n$

end if

end function

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

A CRF is represented through a graph structure $H = (V, E)$, where V is a set of nodes stating random variables, and E a set of edges linking related nodes. Concretely, in the joint object recognition problem, each variable in Y (indeed, each object being categorized) introduces a node in V, and two contextually related variables[2] set an edge in E between their respective nodes. Then, according to the Hammersley-Clifford theorem [35], a number of functions called factors are defined over parts of H, encoding each one a piece of $P(Y|X)$. In this work we rely on two factor types: first order or *unary*, related to nodes, and second order or *pairwise*, associated with edges. The insight behind this is that unary factors encode the likelihood of a variable y_i to be assigned to a certain category l_i given the characterized object x_i, while pairwise factors express the compatibility of two related variables belonging to a certain pair of categories.

Concretely, unary factors $U(\cdot)$ and pairwise factors $I(\cdot)$ are defined by linear classification models as follows:

$$U(y_i, x_i, \boldsymbol{\omega}) = \sum_{l \in L} \delta(y_i = l) \boldsymbol{\omega}_l f(x_i) \tag{2}$$

$$I(y_i, y_j, x_i, x_j, \boldsymbol{\theta}) = \sum_{l_1 \in L} \sum_{l_2 \in L} \delta(y_i = l_1) \delta(y_j = l_2) \boldsymbol{\theta}_{l_1,l_2} g(x_i, x_j) \tag{3}$$

where $f(x_i)$ is the function that computes the vector of unary or local features $\boldsymbol{f}_{x_i u}$ of the object x_i, $g(x_i, x_j)$ provides the pairwise features $\boldsymbol{f}_{x_i x_j p} = (f_{x_i x_j p_1} \ \ldots \ f_{x_i x_j p_q})^\mathsf{T}$, or features that characterize the relation between two objects x_i and x_j, (e.g., perpendicularity, size ratio, etc.), $\boldsymbol{\omega}_l = (\omega_{1,l} \ldots \omega_{m,l})$ and $\boldsymbol{\theta}_{l_1,l_2} = (\theta_{1,l_1,l_2} \ldots \theta_{q,l_1,l_2})$ are vectors of weights associated to the category l and the combination of categories l_1 and l_2 respectively, both learned during the CRF training, and $\delta(y_i = l)$ is the Kronecker delta function.

[2] Two variables y_i and y_j are related if their associated objects x_i and x_j are located close to each other in the scene.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Once these factors have been defined, the computation of $P(Y|X)$ can be expressed by means of log-linear models as:

$$P(Y|X, \boldsymbol{\omega}, \boldsymbol{\theta}) = \frac{1}{Z(X, \boldsymbol{\omega}, \boldsymbol{\theta})} e^{-\epsilon(Y, X, \boldsymbol{\omega}, \boldsymbol{\theta})} \tag{4}$$

where $Z(\cdot)$ is the partition function, which plays a normalizing role so that $\sum_{\xi(Y)} P(Y|X, \boldsymbol{\omega}, \boldsymbol{\theta}) = 1$, with $\xi(Y)$ being a possible assignment to the variables in Y, and $\epsilon(\cdot)$ is the so-called energy function defined as the sum of all the factors codified over the graph:

$$\epsilon(Y, X, \boldsymbol{\omega}, \boldsymbol{\theta}) = \sum_{i \in V} U(y_i, x_i, \boldsymbol{\omega}) + \sum_{(i,j) \in E} I(y_i, y_j, x_i, x_j, \boldsymbol{\theta}) \tag{5}$$

The training phase of the CRF provides the vectors of weights $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ that maximize the following likelihood function:

$$\max_{\boldsymbol{\omega}, \boldsymbol{\theta}} L_P(\boldsymbol{\omega}, \boldsymbol{\theta} : D) = \max_{\boldsymbol{\omega}, \boldsymbol{\theta}} \prod_{d \in D} P(y_d|x_d) \tag{6}$$

where D is the set of all the scenes used for training, x_d is a set containing the characterized objects in the scene $d \in D$, and y_d contains their corresponding ground truth categories.

Solving Equation 6 requires the computation of the partition function, which is infeasible in practice. To face this two approaches arise [59]: the utilization of an alternative objective function, or the estimation of the original one by means of an approximate inference method. In this work, we have opted for the first group of solutions, concretely the alternative, pseudo-likelihood function, which has shown that (given a large number of training samples) the optimization results converge to the ones yielded by the likelihood function [35].

Despite this simplification, the learning process remains complex if the number of considered: categories $|L|$; features used to describe them $\left|\boldsymbol{f}_{x_i u}\right|$; and features characterizing their relations $\left|\boldsymbol{f}_{x_i x_j p}\right|$ is high, which results in a large number of weights as well. On the other hand, it is desirable to account for a

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

comprehensive variety of features in order to properly characterize both objects and relations. In Sec. 4.3.3 we describe our approach to tackle this issue.

4.3.2. Scene Graph Construction and Feature Extraction

Now that the theory behind CRF models has been discussed, we put it into practice by building a scene graph for both the portion of the scene being inspected in each available RGB-D image (recall Figure 3), and any relevant contextual information out of it. For that, first it is added a node for each tracked anchor, then recursively adding nodes for untracked anchors that are closer than a certain context range to a node already in the graph (which can include known objects that are close to currently observed ones, but outside the camera frustum). An edge is added to the graph between each pair of nodes that are within that context range of each other (see Figure 5 on page 21).

Notice that the scene graph corresponds to a CRF graph structure $H = (V, E)$, where V is the set of nodes, each one associated with a random variable from Y representing the category of an object x_i, and E is the set of edges/relations E between those objects.

These objects and relations are subsequently characterized through the vectors of features $\boldsymbol{f}_{x_i u}$ and $\boldsymbol{f}_{x_i x_j p}$ respectively, which are integrated into the CRF model as introduced in Equation 2 and Equation 3 on page 32.

The features are based on the objects' bounding boxes and the boundary and center of the table. The bounding boxes are computed so that their z axis aligns with the metric map frame's z axis, which is pointing upwards. The table boundary is computed from the sensor data and represented as a (not necessarily rectangular or convex) polygon. Figure 8 illustrates the table boundary and some of the features that relate to it.

The unary features used here are listed in Tab. 2. They describe the shape, size and position on the table of an object independent of other objects. This allows the CRF to learn both the typical shape and size of different object categories as well as their typical position on the table: some objects might occur more often near the center of a table, while others are usually near the

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

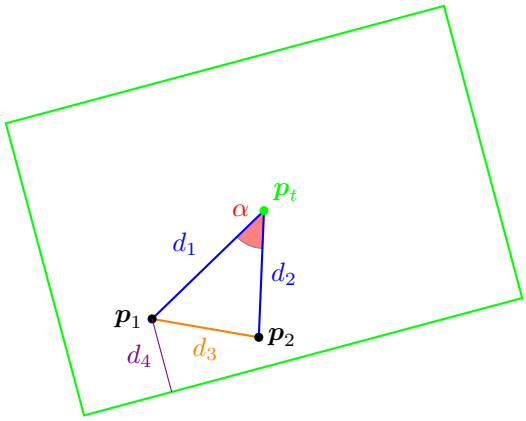


Figure 8: Illustration of some of the CRF unary and pairwise features (Tables 2 and 3).

$\boldsymbol{p}_1$, $\boldsymbol{p}_2$: object centroids

$\boldsymbol{p}_t$: table center

d_1, d_2: distance from $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ respectively to $\boldsymbol{p}_t$ (`table_center_distance`(x_1), `table_center_distance`(x_2))

d_3: horizontal distance between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ (`hdist_centroids`($\mathtt{x}_1, \mathtt{x}_2$))

d_4: distance from $\boldsymbol{p}_1$ to table border (`table_border_distance`($\mathtt{x}_1$))

α: angle between $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_t$ (`table_angle_diff`($\mathtt{x}_1, \mathtt{x}_2$)).

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

table border.

The pairwise features are listed in Tab. 3 and describe the relative size and position on the table of two objects. These features allow the CRF to exploit context between the objects; for example, some objects usually occur close to each other or at the same distance from the table border.

We chose the features to fulfill two criteria. The first criterion is that the features should be continuous/quantitative and not discrete/qualitative. The reason for preferring quantitative over qualitative features is that it avoids hard discrete jumps in the value of the feature near thresholds, which would negatively impact the classification performance and would introduce another set of parameters (namely, the thresholds).

The second criterion is that the features should not rely on any implicit reference poses that arise from peculiarities of how the robot has recorded its data. For example, Anand et al. [2] use the node feature "Distance from the scene boundary". In their system, the scene boundary is implicitly given by the fringe of the area mapped by the robot. Since in their experiments, the robot has always mapped exactly one room, this feature gives the distance to the room's walls; however, this feature is not applicable when only part of a room or multiple rooms are mapped. Another example from Anand et al. [2] is the edge feature "Relative position from camera (in-front-of /behind)". Since their system runs offline on a pre-registered dataset from many camera poses, there is no unique reference camera pose. The paper does not include details, but presumably the "camera pose" here refers to the origin of the registered map, which is usually arbitrarily set to the pose of the first camera frame. This again introduces a dependency on a particularity of the used dataset. Likewise, the features used by Kunze et al. [41] use a reference system that is based on the front-left table corner. This is only meaningful because all of their datasets have been recorded from the "front" of the office desks, i.e., with the monitor in the back and the keyboard closer to the robot.

The only assumption made in the features used in this work is that the sensor data is represented in (or can be transformed into) a coordinate frame

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Table 2: CRF unary features. The variables d_x, d_y, d_z used in the first five features are the dimensions of the object's bounding box; the z axis is oriented upwards. The two features `table_border_distance` and `table_center_distance` are illustrated in Fig. 8.

Feature name	**Definition**	**Description**
`volume`	$d_x d_y d_z$	Volume of the bounding box
`area_h`	$d_x d_y$	Horizontal area of the bounding box
`area_v`	$d_z \sqrt{(d_x)^2 + (d_y)^2}$	Vertical area of the bounding box
`elong_h`	$\max(d_x, d_y) / \min(d_x, d_y)$	Horizontal elongation of the bounding box
`elong_v`	$d_z / \sqrt{(d_x)^2 + (d_y)^2}$	Vertical elongation of the bounding box
`table_border_distance`	— (see Fig. 8)	Distance to the table border polygon
`table_center_distance`	— (see Fig. 8)	Distance to the table center

Table 3: CRF pairwise features for an edge between objects x_1 and x_2 with centroids $\boldsymbol{p}_1 = (p_{x1} \;\; p_{y1} \;\; p_{z1})^\mathsf{T}$ and $\boldsymbol{p}_2 = (p_{x2} \;\; p_{y2} \;\; p_{z2})^\mathsf{T}$. The functions refer to the unary features in Table 2. The three table-related features are illustrated in Fig. 8.

Feature name	**Definition**	**Description**
`bias`	1	Bias term (models co-occurrence probability between object classes)
`volume_ratio`	$\texttt{volume}(x_1) / \texttt{volume}(x_2)$	Ratio of bounding box volumes
`hdist_centroids`	$\sqrt{(p_{x1} - p_{x2})^2 + (p_{y1} - p_{y2})^2}$	Horizontal distance between centroids
`table_border_diff`	$\texttt{table\_border\_distance}(x_1)$ $- \texttt{table\_border\_distance}(x_2)$	Difference of distances to the table border
`table_center_diff`	$\texttt{table\_center\_distance}(x_1)$ $- \texttt{table\_center\_distance}(x_2)$	Difference of distances to the table center
`table_angle_diff`	— (see Fig. 8)	Signed angle between $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and table center

where the z axis points up, which is the case on all ground-based robots. Some of the features are expressed in relation to a table, but that table center and boundary is extracted from the sensor data by our system and is independent from any reference poses. Without a reference pose, it is not directly possible to capture "in front of / behind / left of / right of" relations; however, the `table_center_distance` and `table_border_distance` features capture some of the "in front of / behind" information, and the signed `table_angle_diff` feature approximates the "left of / right of" relation.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

4.3.3. Integrating local object recognition results

Our CRF model can be enriched with the results from any local object recognition method (or a combination of them) able to provide a confidence vector of its results, which enables the usage of methods exploiting specialized feature descriptors. For that, we introduce the confidence vector $\boldsymbol{c}_{x_i}$ of an object x_i into the usual CRF unary factors formulation (see Equation 2) in the following way:

$$U(y_i, x_i, \boldsymbol{\omega}) = \sum_{l \in L} \delta(y_i = l) \boldsymbol{\omega}_l f(x_i) \boldsymbol{c}_{x_i}(l) \tag{7}$$

Thus, the components of the confidence vector play the role of an additional feature, not related to any weight, that have a different value for each object category. This integration also leads to a reduction in the number of weights present during the CRF learning, alleviating the training complexity. In this way, the CRF focuses on exploiting high-level features of objects and relations, and releases the work with specialized feature descriptors to the local method, which modeling into the CRF typically involves a large number of weights.

As an illustrative example, in our experiments we have considered 9 different object categories, 7 unary features, and 6 pairwise ones (see Section 4.3.2). The number of weights associated with features of objects is $\left|\boldsymbol{f}_{x_i u}\right| \cdot |L| = 7 \cdot 9 = 63$, while the number of those related to contextual features is $\left|\boldsymbol{f}_{x_i, x_j p}\right| \cdot |L|^2 = 6 \cdot 81 = 486$, giving a total of 549 weights. In the case of the local object recognition system considered in this work, it represents each object by a set of 10 shape features, each one with a descriptor vector of 45 components, resulting in a total descriptor length of 450. Thus, their modeling into a CRF would suppose the addition of $450 \cdot |L| = 4050$ weights, which clearly would increase the training complexity, even dropping the recognition performance.

4.3.4. Probabilistic inference

Once the CRF model is complete, including: a graph representation $H = (V, E)$, the extracted features of the graph components, and the confidence

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

values from the local object recognition method, a probabilistic inference process over such a CRF performs the joint object recognition. Concretely, it finds an assignment of categories $\hat{Y}$ to the objects in X that maximizes the probability $P(Y|X)$, that is:

$$\hat{Y} = \arg\max_{Y} P(Y|X, \boldsymbol{\omega}, \boldsymbol{\theta}) \tag{8}$$

This issue is commonly referred as the Maximum a Posteriori problem (MAP), which in this work is carried out by the Iterated Conditional Modes method [5], an approximated solution that mitigates the heavy computational burden required by exact approaches. We have made our implementation of the aforementioned training and inference processes available as part of the Undirected Probabilistic Graphical Models in C++ library (UPGMpp), an open source toolkit for efficiently building, training and performing inference over CRFs [58].

4.4. Blackboard Synchronization

The context-aware anchoring system is embedded in the rest of the plan-based robot control architecture via the blackboard (see Section 3). The anchoring system however maintains its own internal world model (see Sec. 4.2) instead of directly storing all anchors on the blackboard for the following reasons: (1) the anchors in the internal world model are a richer representation of the objects (including a probabilistic representation of the possible object types) than the purely symbolic fluents on the blackboard, and (2) the internal anchors have to be updated at a high frequency, while the blackboard is designed to store knowledge over longer time horizons with slower update rates.

In order to make the poses and identities of objects available to other modules (e.g., the planner and plan executor), the internal world model of the anchoring system is continually synchronized with the blackboard. This synchronization happens in two directions, as explained next: (1) reading fluents from the blackboard and (2) writing fluents to the blackboard.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Reading fluents from the blackboard. When starting up, all already existing object-related fluents are retrieved from the blackboard using the following SPARQL query:

```
SELECT DISTINCT ?instance WHERE
{
  ?instance upper:hasBoundingBox ?any .
  ?instance a ?subclass .
  {
    {?subclass rdfs:subClassOf+ race:Kitchenware}
    UNION {?subclass rdfs:subClassOf+ race:PersonalItem}
  }
}
```

This query is also registered with the blackboard, so whenever a new fluent is added to the blackboard that matches this query, the anchoring system is immediately notified.

A sample set of object fluents retrieved by this query is shown in Listing 1 on page 42. Each object is represented by a set of fluents. The main object fluent (here: `mug1`) states the object's category and a list of poses and bounding boxes. Each bounding box fluent specifies the dimensions of the bounding box and its centroid pose. Note that this results in each object having two poses. The reason is that when a CAD model based local object recognition is used, the object's pose relates to the CAD model's reference frame, and the pose of the bounding box is always the bounding box's centroid (see Fig. 9 on page 44 for an illustration). For local object recognition methods that are not based on CAD models, the object's pose and the bounding box's pose coincide.

Writing fluents to the blackboard. When a new object is *discovered*, a corresponding fluent is immediately created on the blackboard. Conversely, when an anchor is *destroyed*, the corresponding fluent is ended immediately.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

When an existing fluent is updated (via *Reacquire* or *Track*), the object's pose is updated as shown in Listing 2 on page 43:

- The existing pose, bounding box and bounding box pose fluents are ended (i.e., the `FinishTime` is set to the current time).
- New pose, bounding box and bounding box pose fluents are created which start at the current time and are open (i.e., the FinishTime is unknown, signified by the special value 0).
- The new fluents are added to the main object fluent (here: `mug1`).

By storing each object pose in a new fluent (instead of simply overwriting the old fluents), the blackboard enables reconstruction of an object's full trajectory over time. This was one of the blackboard's design goals, since this functionality is essential for other tasks within the RACE project (such as learning from experiences).

To avoid overloading the blackboard with minute pose changes at a high frequency, the object poses are only updated if the pose difference is above a certain threshold or after a timeout.

5. System evaluation

To evaluate our system, we collected a dataset of 15 different scenes. After a description of the dataset and the cross validation scheme, we first present a separate evaluation of the object association part of the system (Section 5.3). This is followed by an evaluation of the complete system (Section 5.4).

5.1. Data recording and preparation

In recent years, many RGB-D datasets have become available; a recent survey [22] lists 102 different 3D datasets, with the majority having been recorded with an RGB-D camera, mostly the Microsoft Kinect 1. Some of the best-known datasets are: NYU Depth v2 [66] (the successor of the popular NYU Depth v1 dataset [65]), Berkeley B3DO [31], and SUN3D [77].

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Listing 1 Initial fluents about the object `mug1` from an external knowledge source. Properties not explicitly specified (like `hasRoll`, `hasPitch`) have the default value 0.

```
!Fluent
Class_Instance: [Coffee, mug1]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasPose, Pose, poseMug1_0]
   - [hasBBox, BBox, bBoxMug1_0]
---
!Fluent
Class_Instance: [Pose, poseMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasX, xsd:float, 5.59]
   - [hasY, xsd:float, 10.30]
   - [hasZ, xsd:float, 0.71]
   - [hasYaw, xsd:float, -1.40]
```

```
!Fluent
Class_Instance: [BBox, bBoxMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasPose, Pose, poseBBMug1_0]
   - [hasXSize, xsd:float, 0.070]
   - [hasYSize, xsd:float, 0.094]
   - [hasZSize, xsd:float, 0.084]
---
!Fluent
Class_Instance: [Pose, poseBBMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasX, xsd:float, 5.59]
   - [hasY, xsd:float, 10.29]
   - [hasZ, xsd:float, 0.75]
   - [hasYaw, xsd:float, 0.0]
```

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Listing 2 Updated fluents about the object `mug1` after one observation

```
!Fluent
Class_Instance: [Coffee, mug1]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasPose, Pose, poseMug1_0]
   - [hasBBox, BBox, bBoxMug1_0]
   - [hasPose, Pose, poseMug1_1]
   - [hasBBox, BBox, bBoxMug1_1]
---
!Fluent
Class_Instance: [Pose, poseMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [25357.284, 25357.284]
Properties:    # unchanged
   - [hasX, xsd:float, 5.59]
   - [hasY, xsd:float, 10.30]
   - [hasZ, xsd:float, 0.71]
   - [hasYaw, xsd:float, -1.40]
---
!Fluent
Class_Instance: [BBox, bBoxMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [25357.284, 25357.284]
Properties:    # unchanged
   - [hasPose, Pose, poseBBMug1_0]
   - [hasXSize, xsd:float, 0.070]
   - [hasYSize, xsd:float, 0.094]
   - [hasZSize, xsd:float, 0.084]
---
!Fluent
Class_Instance: [Pose, poseBBMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [25357.284, 25357.284]
Properties:    # unchanged
   - [hasX, xsd:float, 5.59]
   - [hasY, xsd:float, 10.29]
   - [hasZ, xsd:float, 0.75]
   - [hasYaw, xsd:float, 0.0]
```

```
!Fluent
Class_Instance: [Pose, poseMug1_1]
StartTime: [25357.284, 25357.284]
FinishTime: [0.0, 0.0]
Properties:
   - [hasX, xsd:float, 5.63]
   - [hasY, xsd:float, 10.24]
   - [hasZ, xsd:float, 0.79]
   - [hasRoll, xsd:float, 0.015]
   - [hasPitch, xsd:float, 0.001]
   - [hasYaw, xsd:float, 2.186]
---
!Fluent
Class_Instance: [BBox, bBoxMug1_1]
StartTime: [25357.284, 25357.284]
FinishTime: [0.0, 0.0]
Properties:
   - [hasPose, Pose, poseBBMug1_1]
   - [hasXSize, xsd:float, 0.101]
   - [hasYSize, xsd:float, 0.107]
   - [hasZSize, xsd:float, 0.076]
---
!Fluent
Class_Instance: [Pose, poseBBMug1_1]
StartTime: [25357.284, 25357.284]
FinishTime: [0.0, 0.0]
Properties:
   - [hasX, xsd:float, 5.63]
   - [hasY, xsd:float, 10.24]
   - [hasZ, xsd:float, 0.79]
   - [hasRoll, xsd:float, 0.015]
   - [hasPitch, xsd:float, 0.001]
   - [hasYaw, xsd:float, 2.186]
```

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

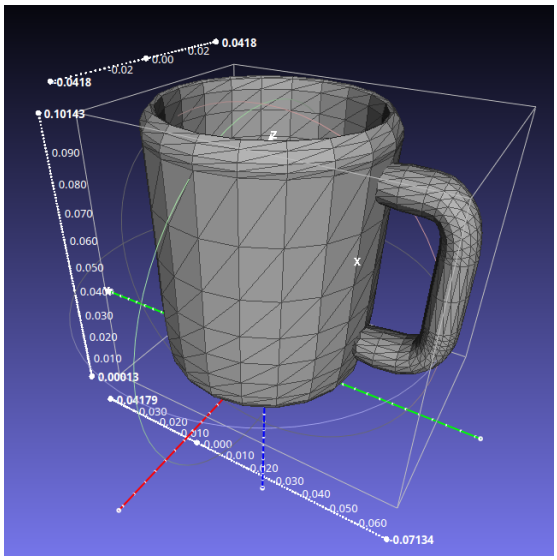


Figure 9: CAD model of the object "mug". The colored axes show the object's reference frame, and the white box is the object's bounding box.

These three datasets are geared towards evaluation of semantic segmentation and labeling algorithms, i.e., they contain per-pixel category labels for each frame, but no bounding box or object pose. The SUN RGB-D [68] dataset addresses this shortcoming by incorporating selected images from the NYU Depth v2, Berkeley B3DO and SUN3D datasets and own images, and adding additional ground truth information, like 3D bounding boxes and object orientation. Also, the SUN RGB-D dataset provides a transformation for each frame that aligns the frame to the gravity direction. Since this information was missing from the original datasets, the annotation was done semi-automatically using the distribution of normals as a heuristic and correcting errors manually.

What is missing from all standard benchmark datasets however is a transformation of each RGB-D frame to a global coordinate frame, not only gravity direction. This is due to the fact that the datasets were recorded using a handheld camera. In robotics, it is commonplace to have access to this information from laser scanner-based localization, so our algorithm uses it to get an initial

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

guess for the position of previously encountered objects. Also, the two local object recognition methods [49, 11] that were integrated with our system only deal with tabletop scenarios, whereas in the standard benchmarking datasets many large-scale structures are labeled (e.g., wall, ceiling, book cabinet). This is also the case in the recent Robot@Home dataset [60], that although includes a global coordinate frame, also contains those structures. For this reason, we decided to record our own dataset using a mobile robot.

This dataset consist of 15 scenes inspected by a robot equipped with a RGB-D camera (Figure 10) driving around a table and turning towards it from different locations. The table contained a number of objects in varying table settings. In total, the dataset contains 1387 seconds of observation and 144 unique objects from 9 categories: sugar pot, milk pot, coffee jug, mug, dish, fork, knife, spoon, and table sign.[3]

Segmentation, tracking and local object recognition was run on the recorded sensor data, and its output (tracked objects and local recognition results) was added to the dataset. Since the objects were observed from multiple perspectives and tracking was lost while the robot was moving from one observation pose to another, the dataset contains more than one track ID for most objects (one for each subsequent observation of the object). Each track ID was manually labeled with the ground truth category of the object it represented. Additionally, all track IDs belonging to the same object were manually grouped together to allow evaluation of the anchoring process. Track IDs that did not correspond to any object on the table (but instead to objects on different tables, pieces of the table itself or other artifacts) were manually removed. In total, out of 432 track IDs, 410 (94.9 %) were associated with true objects, while 22 (5.1 %) were removed as artifacts.

[3] The dataset is available at the following URL: `https://doi.org/10.5281/zenodo.1257047`

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Figure 10: The robot used to record the data set, equipped with an ASUS Xtion Pro Live RGB-D camera (top) that was removed from its casing and a pair of laser scanners for localization, navigation and obstacle avoidance.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

5.2. Cross validation

For evaluation, k-fold cross validation was performed. When choosing the splitting scheme of the data into folds, it is important to account for the fact that the samples (i.e., the RGB-D frames) in the dataset are not independently and identically distributed. Instead, there is a high statistical dependency between samples from each of the 15 scenes, since the configuration of objects within a scene did not change (only the perspective of the robot changed). A random split of the samples into the k folds would result in overfit and an inflated validation score, since the system would be tested on samples that are artificially close to the training samples.

For this reason, a splitting scheme was chosen where iteratively all the samples within one of the 15 scenes were left out of the dataset during training, and those samples were then used for testing. This cross-validation scheme is sometimes called leave-one-label-out[4] [LOLO; 63]. In the neurosciences, it is called leave-one-subject-out [LOSO; 18].

5.3. Object Association

This section evaluates the capabilities of the anchoring process carried out before contextual relations are exploited. This step first computes the similarities among the objects being observed and the set of anchors already present in the system (Section 5.3.1), and then associates these observations to anchors (Section 5.3.2). As illustrated in Figure 3, the outcome of this anchoring is further updated after contextual object recognition is carried out.

5.3.1. Object-Anchor Similarity Computation

We used the dataset to train and evaluate the SVM that computes the similarity between two object observations in order to decide if they are observations of the same object or not (see Section 4.2.1). To transform the dataset

[4]The term "label" in this context denotes the scene to which a sample belongs, not the more common usage of the target class of a sample.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

into training and test data for the SVM, we ran our system while simulating "perfect" object association: instead of using the combination of the SVM and the Hungarian Algorithm for object-anchor data association, we used ground truth to associate new track IDs to the correct anchors. For each combination of new track IDs and untracked anchor, the similarity features were computed, and a sample was added to the transformed dataset. The sample was labeled as "same object" if the association was correct according to ground truth and "different object" otherwise.

Figure 11 shows a scatter plot of the transformed dataset. For visualization purposes, only two similarity features are used in the Figure (*distance* and *diff_object_types (norm)*). As expected, samples labeled "same object" are grouped in the lower right corner: Both distance and object type difference are low. Somewhat surprisingly, the distance between object observations ranges up to 0.3 m; since the objects were not moved during the recording of the dataset, this observed distance is fully caused by localization errors. The figure also shows that the two classes are not linearly separable, and that both the distance between observed positions over time (feature "distance") and similar appearance (feature "diff_object_types") are useful features to correctly associate an object to previous observations.

To evaluate which features yield the best classification results, the SVM was trained on four different sets of features:

(a) distance + diff_object_types (norm): Distance between the centroids of the object observations + norm of the difference vector between the object types. This feature set corresponds to Figure 11.

(b) distance + diff_object_types (vector): Same as (a), but instead of the norm of the object type difference vector, each object type difference is used as a separate feature.

(c) distance + diff_object_types (norm) + additional features: Same as (a), but with additional features (*diff_v_elong, diff_h_elong, diff_v_area, diff_h_area, volume_ratio*).

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

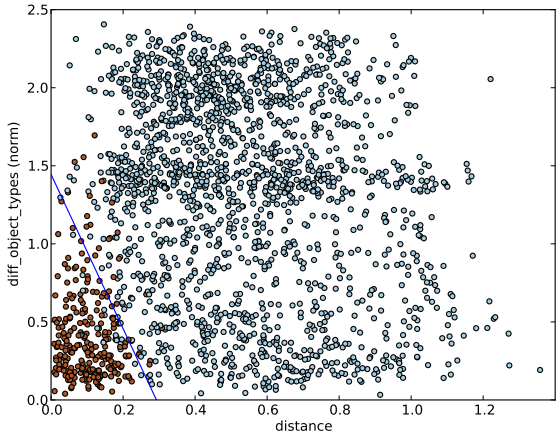


Figure 11: Distribution of samples in the transformed dataset. Red dots: samples labeled "same object", blue dots: samples labeled "different object". For visualization purposes, the only two similarity features used in this figure are euclidean distance between object centroids (x axis) and the norm of the difference vector between the object types (y axis). Blue line: decision boundary learned by the SVM.

(d) distance + diff_object_types (vector) + additional features: Same as (b), but with additional features.

The results are shown in Table 4. Even the smallest set of features (a) already yields good classification results (F_1 score: 0.97), which shows that the distance between objects and the aggregated difference of object types (a) is sufficient information for the classifier to decide whether two objects are the same. Providing additional information (b–d) does not further increase the classification performance significantly. Since simpler models with less parameters are more robust and less prone to overfitting, the simplest model (a) was chosen for the rest of this evaluation. To investigate whether a combination of features that is not included in a–d would yield a better result, Recursive Feature Elimination (RFE) was applied to the full feature set (c). During RFE, the features are removed one-by-one by eliminating the least informative feature. On some classification problems, this increases accuracy by reducing overfit. RFE results in the following ranking of features by importance:

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

1. *distance*
2. *diff_ object_ types (norm)*
3. *diff_ v_ elong*
4. *diff_ v_ area*
5. *diff_ h_ area*
6. *volume_ ratio*
7. *diff_ h_ elong*

None of the iteratively generated feature sets outperformed (a) significantly. This indeed confirms that the feature set consisting of *distance* and *diff_ object_ types (norm)* is the best choice.

5.3.2. Object-Anchor Data Association

To assess the performance of the complete object-anchor data association algorithm that combines the SVM trained in the previous subsection and the Hungarian algorithm, another experiment was performed. We ran the object-anchor data association algorithm (see Section 4.2.2) in its normal mode of operation on the untransformed dataset. For each frame, the Hungarian algorithm first found the globally best assignment between new track IDs and untracked anchors (as determined by the similarity value from the SVM). Next, for each assignment, a threshold on the similarity value was used to determine whether to create a new anchor for an object or whether to associate it to the assigned anchor. The results are shown in Table 5. In 256 out of 272 cases (94,12 %), the algorithm made the correct decision. These numbers do not include the trivial cases where no untracked anchors are present (such as in the beginning of each run of the algorithm), because in this case, the Hungarian algorithm is not needed; instead, a new anchor is created directly for each track ID. If these cases were included, the reported accuracy would be even higher.

Since the decisions reported in Table 5 ultimately depend on the classification made by the SVM, we also show the classification report of the data association process in Table 6, for easy comparison with the SVM results on the transformed data set (Table 4). However, one should note that the two tables

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Table 4: Classification reports of the SVM trained on different sets of features: precision, recall, F_1 score and support (number of elements in the class)

	precision	recall	F_1 score	support
(a) distance + diff_ object_ types (norm)				
different object	0.99	0.98	0.98	1816
same object	0.88	0.90	0.89	249
avg / total	0.97	0.97	0.97	2065
(b) distance + diff_ object_ types (vector)				
different object	0.99	0.98	0.99	1816
same object	0.89	0.92	0.90	249
avg / total	0.98	0.98	0.98	2065
(c) distance + diff_ object_ types (norm) + additional features				
different object	0.99	0.98	0.98	1816
same object	0.86	0.91	0.89	249
avg / total	0.97	0.97	0.97	2065
(d) distance + diff_ object_ types (vector) + additional features				
different object	0.99	0.98	0.99	1816
same object	0.88	0.92	0.90	249
avg / total	0.98	0.98	0.98	2065

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Table 5: Decisions made by the data association process when using the SVM from Table 4(a). Row labels: Whether in the best assignment found by the Hungarian algorithm, the track ID and the assigned untracked anchor corresponded to *different objects* or the *same object* according to ground truth. Column labels: Whether the algorithm decided to create a new anchor for the track ID or whether to associate it to the assigned anchor, based on the similarity rating from the SVM. Correct decisions are highlighted in **bold**. Total accuracy is 94,12 %.

	created new anchor	associated to existing anchor	total
different object	**28 (84,8 %)**	5 (15,2 %)	33 (100 %)
same object	11 (4,6 %)	**228 (95,4 %)**	239 (100 %)

Table 6: Classification report of the data association process when using the SVM from Table 4(a). The data in this table corresponds to Table 5.

	precision	recall	F_1 score	support
different object	0.72	0.85	0.78	33
same object	0.98	0.95	0.97	239
avg / total	0.95	0.94	0.94	272

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

are not directly comparable: Table 4 contains multiple entries for each new track ID (one for each possible assignment of the track ID to an untracked anchor), whereas Table 6 only has one entry for the best assignment. This results in two opposing effects on the classification accuracy. On the one hand, the Hungarian algorithm ensures that in those cases where the SVM would accept two or more possible matches, only the best match is counted, thereby increasing accuracy on the hard-to-decide cases near the class boundary and improving the accuracy of the category "same object". On the other hand, this step also eliminates all the easy-to-decide cases of different objects far away from the class boundary (the ones in the top right corner of Figure 11), which results in a lower reported accuracy for the class "different object". This is also evident in the fact that in Table 6, the support of class "different object" is much lower than in Table 4. Also, the transformed dataset used "perfect object association" (from ground truth), whereas in the actual data association, errors can propagate: When a track ID is assigned to the wrong untracked anchor, that anchor is not available for the correct assignment later on. However, even given those interactions between the Hungarian algorithm and the SVM, the object-anchor data association still reaches an F_1 score of 0.94 (compared to 0.97 for the SVM alone on the transformed dataset), which demonstrates that our data association step is able to correctly assign almost all track IDs to the correct anchor.

5.4. Complete System

The collected dataset from table-top settings has two challenging characteristics for both local and contextual object recognition methods, which further motivates the utilization of the proposed system. On the one hand, the small objects (fork, knife and spoon) only produce a small set of points in the captured point clouds (or pixels in the RGB-D images); they are almost the same size and contain reflective parts, so only the handle is actually visible. This combination makes them hard to be distinguished based only on local features. On the other hand, the robot normally sees only a part of the scene, which means that the full object context information is not available from the current sensor data (see

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

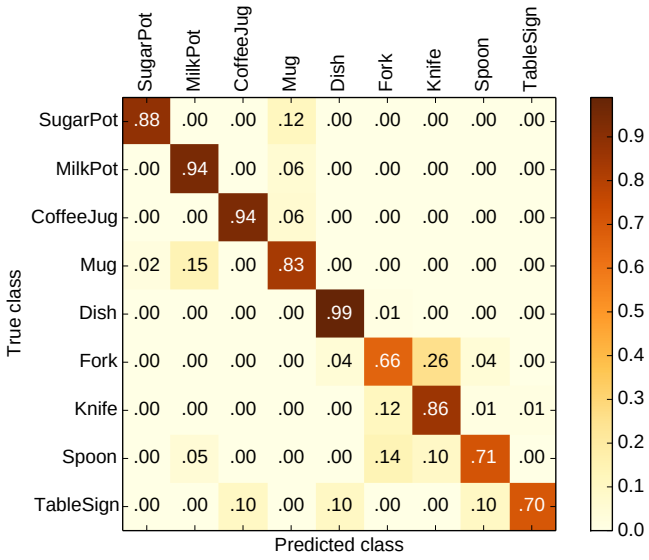


Figure 12: Confusion matrix of the combined method (CRF+A+Loc in Table 7) – classification accuracy: 86.82 %.

Figure 5).

Since an object is assigned a new track id when it leaves and then re-enters the robot's field of view, the number of tracks (387) is higher than the number of unique objects (144). The predicted category was determined by the class that was reported most often by the CRF for one particular track ID.

The results are summarized in Table 7. As expected, recognizing small objects yields poor accuracy results when using only the local object recognition method. However, our combined system exploits context to achieve a significant improvement (29.22 % increase in accuracy). For the remaining six objects, the local object recognition method already delivers a high accuracy (91.15 %), so the improvement from exploiting context is not as large (2.0 % increase in accuracy). This shows that exploiting context is most useful in those cases where objects are ambiguous and hard to distinguish based on local appearance alone, but also improves accuracy in the other cases. The results also demonstrate the performance boost by using context with objects outside the current field

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Table 7: Classification accuracy on sub-datasets. *Loc*: local object recognition only [base line; 49]; *CRF+A*: CRF only, but with objects outside the field of view supplied by Anchoring; *CRF+Loc*: CRF integrated with local object recognition; *CRF+A+Loc*: complete, proposed system (CRF, anchoring, local object recognition)

	Loc	CRF+A	CRF+Loc	CRF+A+Loc
small objects (fork, knife, spoon)	46.32 %	75.00 %	72.58 %	75.54 %
other objects	91.15 %	90.73 %	94.38 %	93.15 %
total	76.64 %	85.05 %	85.43 %	86.82 %

of view (CRF+A, Sec. 4.2) and integrating local object recognition results into the CRF (CRF+Loc, Section 4.3). In total, the complete combined system (CRF+A+Loc) achieved an increase of 10.18 % in accuracy over the employed local object recognition method [49]. Figure 12 shows the aggregated confusion matrix yielded by the proposed system on the dataset.

Notice that the achieved accuracy partially depends on the success of the local object recognition method, as mentioned in Section 4.1. Since this method could be replaced by any other method providing the same output, the important thing here is not the reported figures about success percentages, but the fact that the exploitation of contextual relations largely improves it performance.

Regarding computational costs, training, which has to be completed only once, took on average 1421.5 s per fold. The runtime of feature extraction was 0.064 s, and for MAP inference 0.005 s per scene graph (excluding time for the local object recognition). All experiments have been performed on a standard laptop (2.6 GHz Core i7 CPU, 8 GB RAM).

6. Conclusions

This work has presented an anchoring system able to exploit the contextual relations of the objects in the scene to achieve more coherent and stable results, aiming to build suitable representations of the robot workspace. This is a critical aspect of these systems when running in robotic architectures, since a wrong linking between an object and its category would end up with failures, for exam-

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

ple, during task planing or execution. To achieve that we rely on a probabilistic, context-based recognition method based on Conditional Random Fields (CRF). These Probabilistic Graphical Models are typically used to model and exploit the relations in a given scene but, in combination with anchoring, here it is also possible to consider nearby objects beyond the field of view of the sensor, which also improves its performance. With this combination of techniques we reach an *online* operation, the system being able to continually process single frames, a requirement for most plan-based robot control systems.

The proposed system also incorporates in the CRF formulation the output of a local object recognition method, which individually classifies objects by providing a vector of confidence values about their belonging to the considered categories. Although we have described this method as a part of the anchoring system, it can be replaced by any other yielding those confidence values.

The usefulness of the system has been demonstrated in the RACE project, where it was employed for anchoring object symbols used by the planner, execution monitor and human-robot interaction to object perceptions from 3D sensor data produced by an object recognition system.

To evaluate the particularities of this proposal we collected a dataset of 15 scenes with table-top settings. This dataset is specially challenging for both local and contextual object recognition methods, so it results in a good testbed for the system. The conducted experiments assessed both the object association and object recognition capabilities of the system. For the first one, the results reported a precision of 0.95, a recall of 0.94, and a F1-score of 0.94. The figures provided for the recognition were also positive: a 86.82% of success. Notice that the CRF-based recognition process exploits the output of both the local object recognition method and an initial anchoring process. The utilization of these sources of information has proven to be valuable, getting a higher success than the CRF configurations that do not employ that. These results also show the benefits of exploiting context in comparison with a local recognition method. The computational costs analysis reported a runtime for the feature extraction of 0.064s on average, and for CRF inference of 5ms.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

Currently we are studying how to exploit contextual relations just before the initial anchoring process, which will made the system output even more coherent and robust. Additionally we plan to exploit the world model representation for efficiently performing robotic tasks, like object search.

Acknowledgments

This work is supported by the European projects RACE (FP7-ICT-2011-7, grant agreement number 287752), MoveCare (H2020-ICT-2016-1, grant agreement number 732158), the Spanish grant program FPU-MICINN 2010 and the PROMOVE project (ref:DPI2014-55826-R).

[1] Marina Alberti, John Folkesson, and Patric Jensfelt. Relational approaches for joint object classification and scene similarity measurement in indoor environments. In *AAAI Spring Symposium, Qualitative Representations for Robots*, Palo Alto, California, March 2014.

[2] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually guided semantic labeling and search for three-dimensional point clouds. *Int. J. Rob. Res.*, 32(1):19–34, 2013. doi: 10.1177/0278364912461538.

[3] Yaakov Bar-Shalom, Peter K. Willett, and Xin Tian. *Tracking and Data Fusion – A Handbook of Algorithms.* YBS Publishing, 2011. ISBN 9780964831278.

[4] Michael Beetz, Ferenc Balint-Benczedi, Nico Blodow, Daniel Nyga, Thiemo Wiedemeyer, and Zoltan-Csaba Marton. RoboSherlock: Unstructured information processing for robot perception. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 1549–1556. IEEE, 2015. doi: 10.1109/ICRA.2015.7139395.

[5] J. Besag. On the statistical analysis of dirty pictures. *Royal Statistical Society*, Series B(2):259–302, 1986.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

[6] J.L. Blanco, J. González, and J.-A. Fernández-Madrigal. Subjective local maps for hybrid metric-topological SLAM. *Robotics and Autonomous Systems*, 57(1):64–74, 2009. ISSN 0921-8890. doi: 10.1016/j.robot.2008.02.002.

[7] Nico Blodow. *Managing Belief States for Service Robots*. Dissertation, Technische Universität München, München, 2014.

[8] Nico Blodow, Dominik Jain, Zoltan-Csaba Marton, and Michael Beetz. Perception and probabilistic anchoring for dynamic world state logging. In *Humanoids*, pages 160–166. IEEE, 2010. ISBN 978-1-4244-8688-5. doi: 10.1109/ICHR.2010.5686341.

[9] H. A. P. Blom and E. A. Bloem. Interacting multiple model joint probabilistic data association, avoiding track coalescence. In *Proc. IEEE Conf. Decision Control*, volume 3, pages 3408–3415, December 2002.

[10] Sebastian Blumenthal, Herman Bruyninckx, Walter Nowak, and Erwin Prassler. A scene graph based shared 3D world model for robotic applications. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 453–460. IEEE, 2013. doi: 10.1109/ICRA.2013.6630614.

[11] Matei T. Ciocarlie, Kaijen Hsiao, Edward Gil Jones, Sachin Chitta, Radu Bogdan Rusu, and Ioan Alexandru Sucan. Towards reliable grasping and manipulation in household environments. In Oussama Khatib, Vijay Kumar, and Gaurav S. Sukhatme, editors, *Experimental Robotics - The 12th International Symposium on Experimental Robotics, ISER 2010, December 18-21, 2010, New Delhi and Agra, India*, volume 79 of *Springer Tracts in Advanced Robotics*, pages 241–252. Springer, 2010. doi: 10.1007/978-3-642-28572-1_17.

[12] Silvia Coradeschi and Alessandro Saffiotti. An introduction to the anchoring problem. *Robot. Auton. Syst.*, 43(2-3):85–96, 2003.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

[13] S.K. Divvala, D. Hoiem, J.H. Hays, A.A. Efros, and M. Hebert. An empirical study of context in object detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1271–1278, June 2009.

[14] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 2650–2658, Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.304.

[15] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin A. Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust RGB-D object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, pages 681–687. IEEE, 2015. doi: 10.1109/IROS.2015.7353446.

[16] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, June 1987. ISSN 0882-4967. doi: 10.1109/JRA.1987.1087096.

[17] Jos Elfring, S. van den Dries, M. J. G. van de Molengraft, and Maarten Steinbuch. Semantic world modeling using probabilistic multiple hypothesis anchoring. *Robotics and Autonomous Systems*, 61(2):95–105, 2013. doi: 10.1016/j.robot.2012.11.005.

[18] Michael Esterman, Benjamin J. Tamber-Rosenau, Yu-Chin Chiu, and Steven Yantis. Avoiding non-independence in fMRI data analysis: Leave one subject out. *NeuroImage*, 50(2):572–576, 2010. doi: 10.1016/j.neuroimage.2009.10.092.

[19] Thomas Fäulhammer, Michael Zillich, Johann Prankl, and Markus Vincze. A multi-modal RGB-D object recognizer. In *23rd International Conference*

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016, pages 733–738. IEEE, 2016. doi: 10.1109/ICPR.2016.7899722.

[20] Matthias Fichtner. *Anchoring symbols to percepts in the fluent calculus*. PhD thesis, Dresden University of Technology, 2009.

[21] Matthias Fichtner. Anchoring symbols to percepts in the fluent calculus - A general approach to the symbol anchoring problem of cognitive robots. *KI*, 25(1):77–80, 2011. doi: 10.1007/s13218-010-0051-1.

[22] Michael Firman. RGBD datasets: Past, present and future. In *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*, pages 661–673. IEEE Computer Society, 2016. doi: 10.1109/CVPRW.2016.88.

[23] Carolina Galleguillos and Serge Belongie. Context based object categorization: A critical survey. *Comput. Vis. Image Underst.*, 114(6):712–722, June 2010. ISSN 1077-3142. doi: 10.1016/j.cviu.2010.02.004.

[24] Carolina Galleguillos, Andrew Rabinovich, and Serge J. Belongie. Object categorization using co-occurrence, location and appearance. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society, 2008. doi: 10.1109/CVPR.2008.4587799.

[25] Guglielmo Gemignani, Roberto Capobianco, Emanuele Bastianelli, Domenico Daniele Bloisi, Luca Iocchi, and Daniele Nardi. Living with robots: Interactive environmental knowledge acquisition. *Robot. Auton. Syst.*, 78:1–16, 2016. doi: 10.1016/j.robot.2015.11.001.

[26] Martin Günther, Thomas Wiemann, Sven Albrecht, and Joachim Hertzberg. Model-based furniture recognition for building semantic object maps. *Artif. Intell.*, 247:336–351, June 2017. doi: 10.1016/j.artint.2014.12.007. Available online: Jan 23, 2014.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

[27] A. Hermans, G. Floros, and B. Leibe. Dense 3D semantic mapping of indoor scenes from RGB-D images. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2631–2638, May 2014. doi: 10.1109/ICRA.2014.6907236.

[28] Joachim Hertzberg, Jianwei Zhang, Liwei Zhang, Sebastian Rockel, Bernd Neumann, Jos Lehmann, Krishna S. R. Dubba, Anthony G. Cohn, Alessandro Saffiotti, Federico Pecora, Masoumeh Mansouri, Štefan Konečný, Martin Günther, Sebastian Stock, Luis Seabra Lopes, Miguel Oliveira, Gi Hyun Lim, Hamidreza Kasaei, Vahid Mokhtari, Lothar Hotz, and Wilfried Bohlken. The RACE project. *KI - Künstliche Intelligenz*, 28(4):297–304, 2014. ISSN 0933-1875. doi: 10.1007/s13218-014-0327-y.

[29] Javier Hidalgo Carrió, Sascha Arnold, Arne Böckmann, Anna Born, Raúl Domínguez, Daniel Hennes, Christoph Hertzberg, Janosch Machowinski, Jakob Schwendner, Yong-Ho Yoo, and Frank Kirchner. EnviRe - environment representation for long-term autonomy. In *ICRA Workshop on AI for Long-term Autonomy*, Stockholm, May 2016.

[30] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. *Int. J. Comput. Vision*, 80(1):3–15, 2008. doi: 10.1007/s11263-008-0137-5.

[31] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3d object dataset: Putting the kinect to work. In Andrea Fossati, Juergen Gall, Helmut Grabner, Xiaofeng Ren, and Kurt Konolige, editors, *Consumer Depth Cameras for Computer Vision, Research Topics and Applications*, Advances in Computer Vision and Pattern Recognition, pages 141–165. Springer, 2013. doi: 10.1007/978-1-4471-4640-7_8.

[32] S. Hamidreza Kasaei, Miguel Oliveira, Gi Hyun Lim, Luís Seabra Lopes, and Ana Maria Tomé. Interactive open-ended learning for 3D object recog-

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

nition: An approach and experiments. *J. Intell. Robot. Syst.*, 2015. doi: 10.1007/s10846-015-0189-z.

[33] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. *CoRR*, abs/1607.06038, 2016.

[34] Salman Hameed Khan, Mohammed Bennamoun, Ferdous Sohel, and Roberto Togneri. Geometry driven semantic labeling of indoor scenes. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 679–694, Zurich, Switzerland, September 2014. Springer International Publishing. ISBN 978-3-319-10590-1.

[35] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[36] Hema Swetha Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3D point clouds for indoor scenes. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *25th Annual Conference on Neural Information Processing Systems (NIPS-2011)*, pages 244–252, Granada, Spain, December 2011.

[37] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robot. Auton. Syst.*, 66:86–103, 2015. doi: 10.1016/j.robot.2014.12.006.

[38] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 954–962, 2015.

[39] Alexander Krull, Eric Brachmann, Sebastian Nowozin, Frank Michel, Jamie Shotton, and Carsten Rother. PoseAgent: Budget-constrained 6D object

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

pose estimation via reinforcement learning. In *2017 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, 2017.

[40] H. W. Kuhn. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.*, 2(1-2):83–97, 1955. ISSN 1931-9193. doi: 10.1002/nav.3800020109.

[41] Lars Kunze, Chris Burbridge, Marina Alberti, Akshaya Tippur, John Folkesson, Patric Jensfelt, and Nick Hawes. Combining top-down spatial reasoning and bottom-up object class recognition for scene understanding. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pages 2910–2915. IEEE, 2014. doi: 10.1109/IROS.2014.6942963.

[42] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *IEEE International Conference on Robotics and Automation, ICRA2011, Shanghai, China, 9-13 May 2011*, pages 1817–1824. IEEE, 2011. doi: 10.1109/ICRA.2011.5980382.

[43] John J. Leonard and Richard J. Rikoski. *Experimental Robotics VII*, chapter Incorporation of Delayed Decision Making into Stochastic Mapping, pages 533–542. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-45118-1. doi: 10.1007/3-540-45118-8_53.

[44] Gi Hyun Lim, Kun Woo Kim, Hyowon Suh, Il Hong Suh, and Michael Beetz. Knowledge-based incremental Bayesian learning for object recognition. In *Autonomous Learning workshop, ICRA 2013*, 2013.

[45] Lingni Ma, Jörg Stückler, Christian Kerl, and Daniel Cremers. Multi-view deep learning for consistent semantic mapping with RGB-D cameras. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pages 598–605. IEEE, 2017. doi: 10.1109/IROS.2017.8202213.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

[46] John McCormac, Ankur Handa, Andrew J. Davison, and Stefan Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, Singapore, May 2017. IEEE. doi: 10.1109/ICRA.2017.7989538.

[47] Andreas C. Müller and Sven Behnke. Learning depth-sensitive conditional random fields for semantic segmentation of RGB-D images. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6232–6237. IEEE, May 2014.

[48] Aude Oliva and Antonio Torralba. The role of context in object recognition. *Trends Cogn. Sci.*, 11(12):520–527, 2007. ISSN 1364-6613. doi: DOI:10.1016/j.tics.2007.09.009.

[49] Miguel Oliveira, Gi H Lim, Luis Seabra Lopes, Hamidreza Kasaei, Ana Tome, and Aneesh Chauhan. A perceptual memory system for grounding semantic representations in intelligent service robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, Illinois, 2014. IEEE.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[51] A. Ranganathan, E. Menegatti, and F. Dellaert. Bayesian inference in the space of topological maps. *IEEE Transactions on Robotics*, 22(1):92–107, February 2006. ISSN 1552-3098.

[52] D. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automat. Contr.*, 24:843–854, December 1979.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

[53] Emilio Remolina and Benjamin Kuipers. Towards a general theory of topological maps. *Artificial Intelligence*, 152(1):47–104, January 2004. ISSN 0004-3702. doi: 10.1016/S0004-3702(03)00114-0.

[54] Xiaofeng Ren, Liefeng Bo, and D. Fox. RGB-(D) scene labeling: Features and algorithms. In *CVPR*, pages 2759–2766, June 2012. doi: 10.1109/CVPR.2012.6247999.

[55] Sebastian Rockel, Bernd Neumann, Jianwei Zhang, Krishna S. R Dubba, Anthony G. Cohn, Štefan Konečný, Masoumeh Mansouri, Federico Pecora, Alessandro Saffiotti, Martin Günther, Sebastian Stock, Joachim Hertzberg, Ana Maria Tomé, Armando J. Pinho, Luís Seabra Lopes, Stephanie von Riegen, and Lothar Hotz. An ontology-based multi-level robot architecture for learning from experiences. In *Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium*, Stanford, USA, March 2013.

[56] José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. Mobile robot object recognition through the synergy of probabilistic graphical models and semantic knowledge. In *ECAI Workshop on Cognitive Robotics (CogRob)*, 2014.

[57] José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. Exploiting semantic knowledge for robot object recognition. *Knowl.-Based Syst.*, 86:131–142, 2015. doi: 10.1016/j.knosys.2015.05.032.

[58] José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. UPGMpp: a software library for contextual object recognition. In *3rd Workshop on Recognition and Action for Scene Understanding (REACTS)*, 2015.

[59] José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. A survey on learning approaches for Undirected Graphical Models. application to scene object recognition. *Int. J. Approx. Reason.*, 83(Supplement C):434 – 451, 2017. ISSN 0888-613X. doi: 10.1016/j.ijar.2016.10.009.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

[60] José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. Robot@Home, a robotic dataset for semantic mapping of home environments. *I. J. Robotics Res.*, 36(2):131–141, 2017. doi: 10.1177/0278364917695640.

[61] José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. Building multiversal semantic maps for mobile robot operation. *Knowl.-Based Syst.*, 119(Supplement C):257–272, 2017. ISSN 0950-7051. doi: 10.1016/j.knosys.2016.12.016.

[62] Max Schwarz, Anton Milan, Arul Selvam Periyasamy, and Sven Behnke. Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter. *I. J. Robotics Res.*, 0(0):3–15, 2017. doi: 10.1177/0278364917713117.

[63] scikit-learn documentation. Cross-validation: evaluating estimator performance, 2018.

[64] Jamie Shotton, John M. Winn, Carsten Rother, and Antonio Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. J. Comput. Vision*, 81(1):2–23, 2009. doi: 10.1007/s11263-007-0109-1.

[65] Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. In *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, pages 601–608. IEEE, 2011. doi: 10.1109/ICCVW.2011.6130298.

[66] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In Andrew W. Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*, volume 7576 of *Lecture Notes in Computer Science*, pages 746–760. Springer, 2012. doi: 10.1007/978-3-642-33715-4_54.

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

[67] Richard Socher, Brody Huval, Bharath Putta Bath, Christopher D. Manning, and Andrew Y. Ng. Convolutional-recursive deep learning for 3d object classification. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 665–673, 2012.

[68] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 567–576. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298655.

[69] J. Stückler, N. Biresev, and S. Behnke. Semantic mapping using object-class segmentation of RGB-D images. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3005–3010, October 2012. doi: 10.1109/IROS.2012.6385983.

[70] Jörg Stückler, Benedikt Waldvogel, Hannes Schulz, and Sven Behnke. Dense real-time mapping of object-class semantics from RGB-D video. *J. Real-Time Image Processing*, 10(4):599–609, 2015. doi: 10.1007/s11554-013-0379-5.

[71] Akshaya Thippur, Chris Burbridge, Lars Kunze, Marina Alberti, John Folkesson, Patric Jensfelt, and Nick Hawes. A comparison of qualitative and metric spatial relation models for scene understanding. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1632–1640. AAAI Press, 2015.

[72] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

navigation. *Artificial Intelligence*, 99(1):21–71, 1998. ISSN 0004-3702. doi: 10.1016/S0004-3702(97)00078-7.

[73] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127, September 2003. ISSN 0929-5593. doi: 10.1023/A:1025584807625.

[74] Antonio B. Torralba, Kevin P. Murphy, William T. Freeman, and Mark A. Rubin. Context-based vision system for place and object recognition. In *ICCV*, pages 273–280. IEEE Computer Society, 2003. ISBN 0-7695-1950-4.

[75] J.P.C. Valentin, S. Sengupta, J. Warrell, A. Shahrokni, and P.H.S. Torr. Mesh based semantic modelling for indoor and outdoor scenes. In *CVPR*, pages 2067–2074, June 2013. doi: 10.1109/CVPR.2013.269.

[76] Yu Xiang, Xiangdong Zhou, Zuotao Liu, Tat-Seng Chua, and Chong-Wah Ngo. Semantic context modeling with maximal margin conditional random fields for automatic image annotation. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 3368–3375. IEEE Computer Society, 2010. doi: 10.1109/CVPR.2010.5540015.

[77] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1625–1632. IEEE Computer Society, 2013. doi: 10.1109/ICCV.2013.458.

[78] Xuehan Xiong and Daniel Huber. Using context to create semantic 3D models of indoor environments. In *British Machine Vision Conference (BMVC)*, September 2010.

[79] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge. In *2017 IEEE*

Accepted author version. Final version published in Robotics and Autonomous Systems
https://doi.org/10.1016/j.robot.2018.08.016

International Conference on Robotics and Automation (ICRA), Singapore, 29 May - 3 June 2017. IEEE, 2017. doi: 10.1109/ICRA.2017.7989165.

[80] Jianguo Zhang, Marcin Marszalek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *Int. J. Comput. Vision*, 73(2): 213–238, 2007. doi: 10.1007/s11263-006-9794-4.