Towards Deliberative Active Perception using Persistent Memory

Tim Niemueller*, Nichola Abdo†, Andreas Hertle‡, Gerhard Lakemeyer*, Wolfram Burgard†, Bernhard Nebel‡
* Knowledge-based Systems Group, RWTH Aachen University, Germany
†Autonomous Intelligent Systems, University of Freiburg, Germany
‡Foundations of Artificial Intelligence, University of Freiburg, Germany

***Abstract*— Task coordination for autonomous mobile service robots typically involves a substantial amount of background knowledge and explicit action sequences to acquire the relevant information nowadays. We strive for a system which, given a task, is capable of reasoning about task-relevant knowledge to automatically determine whether that knowledge is sufficient. If missing or uncertain, the robot shall decide autonomously on the actions to gain or improve that knowledge. In this paper we present our baseline system implementing the foundations for these capabilities. The robot has to analyze a tabletop scene and increase its object type confidence. It plans motions to observe the scene from multiple perspectives, combines the acquired data, and performs a recognition step on the merged input.**

I. Introduction

For flexible and adaptive autonomous mobile robots, it is crucial to develop systems that require little *a priori* knowledge to operate in new and dynamic environments. Hybrid reasoning, combining symbolic task and geometric motion planning or more generally combining qualitative and quantitative aspects of knowledge representation and reasoning, plays an important role in this context.

In our project, we strive to combine these aspects into a coherent system that will allow the robot to plan not only the sequence of locomotion and manipulation actions that is necessary to achieve the physical goals (e.g. moving an object or turning on an appliance), but also to reason on the information that is necessary to do so. In other words, we want to make information acquisition an integrative part of the overall task execution to enable the robot to actively explore what it needs to solve a task.

Starting out from symbolic approaches like classical planning on PDDL-defined domains and problems [1] and reasoning with projection using GOLOG [2] as a modeling language, research questions that follow are:

- how to represent quantitative data like uncertainty or object positions in the knowledge base?
- how to represent and reason about incomplete knowledge using an explicit epistemic state?
- when to query geometric planners without incurring an infeasible overhead?
- whether and how to acquire task-relevant knowledge?

There are only a few systems that approach all of these (or similar) aspects in a holistic robotic system. KnowRob [3] is an example which focuses on background knowledge and its use, while we present our baseline system for active sensing.

The particular issue we address in this paper is the frequent insufficiency of sensor data and uncertainty about extracted information if a scene is only viewed from a single perspective. We describe our baseline system that is capable of active perception, i.e. taking action to acquire the needed information with sufficient certainty. For now, we focus on analyzing a tabletop scene from varying perspectives. The required building blocks on a PR2 robot are the task planner that generates the sequence of observation points, database recording facilities, multi-perspective point cloud merging to combine data from multiple perspectives, and a recognition module that can determine confidences per object and type. The system has been created with integration points for future extensions towards deliberative hybrid reasoning for mobile manipulation tasks. The most important future extension will be an epistemic state reasoning component, which explicitly represents and reasons about the robot's belief of the world and whether and how to acquire more information. Further development will also improve data fusion and object recognition approaches, and consider manipulation actions to augment perception results.

To render the problem feasible for now, we specified the domain as tabletop scenes in household environments. The tasks are to either identify all objects on a table with high confidence, or to search for a particular object. These constraints allow us to reuse as much existing prior work as possible regarding tasks that are required but not the core of our work, like perception or motion planning.

The rest of the paper is organized as follows. In Section II we present related work and give some background information about our system. We then describe the action planning component in Section III. The automated data recording is explained in Section IV. In Section V we detail the multi-perspective point cloud merging and object detection and recognition, followed by a description of our experiments in Section VI. We conclude in Section VII.

II. Related Work and Background

As we are presenting an integrated robot system, the body of related work is vast. Hence we chose works related to particularly important aspects of our system.

Robot Database: In our previous work, we proposed a method to store any and all data acquired, processed, and communicated between components to a database [4]. We defined certain criteria for storage systems and identified MongoDB [5] as a suitable system. As an example, the database is applied in fault analysis. Mason and Marthi [6]

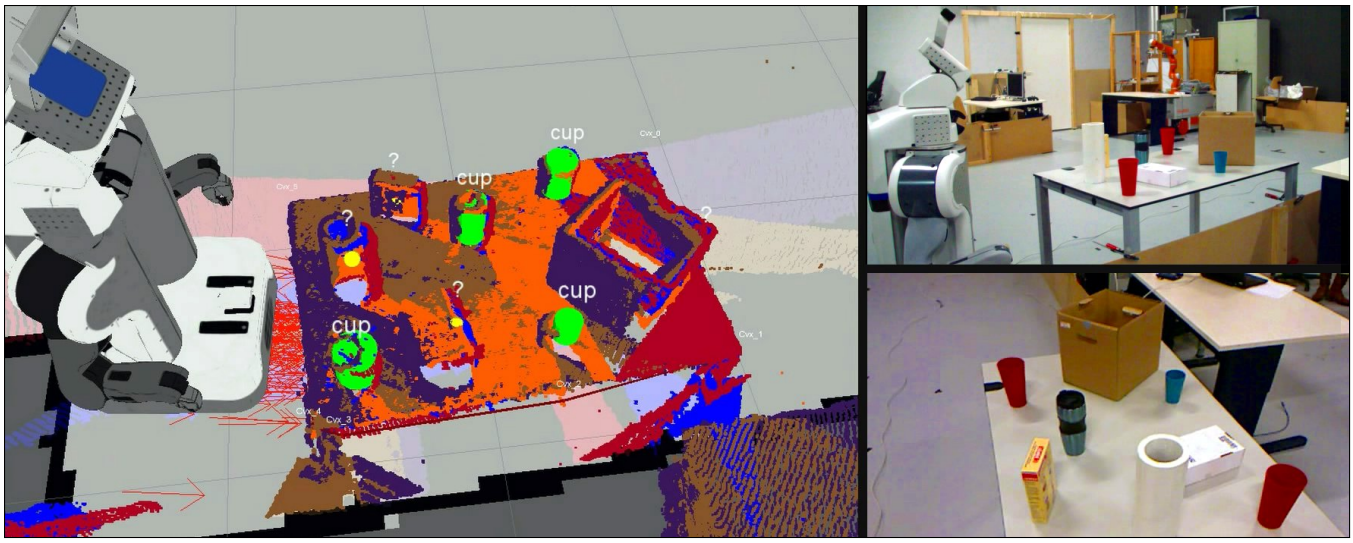


Fig. 1. *Experimental setup* The top right image shows our experimental setup for object recognition, with the corresponding perspective of the robot's camera depicted in the lower right corner (cropped). The left image visualizes the object classification results after merging the point clouds from the different perspectives according to our approach, where detected cups are depicted by green cylinders.

employ MongoDB to record objects in terms of planes and clusters for semantic querying and change detection.

Perception: In our work, we use the Point Cloud Library (PCL) [7] for point cloud merging, alignment, and processing to detect objects. It provides the required basic operations like plane extraction, iterative closest point alignment, and clustering. A real-time approach for generating dense, volumetric reconstruction is KinectFusion [8]. It combines depth information and tracks the sensor's position on a frame-by-frame basis to create a surface model for scene analysis. Compared to our approach it produces a model of the environment with higher accuracy, but at vastly increased computational effort and power consumption, which is a concern in particular on a mobile robot.

Active perception: Several approaches address the problem of active perception for service robots. Stampfer *et al.* [9] presented an approach to classify objects on a table based on features obtained from a Kinect camera. Individual object classification is refined from additional viewpoints selected based on the expected detection quality. More recently, the authors extended their approach and presented a system for object recognition and pose estimation by fusing the results for multiple algorithms in a probabilistic framework [10]. This improves results in many cases where a system based on a single detection algorithm would fail. In similar work, Eidenberger and Scharinger [11] developed a system for probabilistic active perception planning. A POMDP planner computes the next best view pose in order to improve the hypotheses for objects in the scene, which consist of an object class and pose information. Similarly, Aydemir *et al.* [12] present an approach relying on spatial relations between objects to select search strategies when searching for a particular object. These spatial relations, e.g. *in* or *on*, describe possible object locations in the scene. An MDP solver then evaluates this *a priori* knowledge and chooses appropriate sensing actions and strategies to find the desired object. In our approach, we combine percepts from multiple viewpoints to build a better representation of the scene and improve the quality of the object classification.

Hybrid reasoning: Kaelbling *et al.* present a hierarchical approach to integrate task and motion planning [13]. By committing early to choices made by the planner, the plan length is reduced, in turn decreasing the amount of planning time exponentially. Geometric suggesters compute numerical values for operators during the planning process, therefore eliminating the necessity of prior discretization of continuous geometry. Wolfe *et al.* propose hierarchical task networks for robotic manipulation tasks [14]. These HTNs allow symbolic reasoning on the higher levels, while integrating geometric planners at the lowest level, producing kinematic feasible high quality plans. Furthermore, the computed kinematic trajectories are reused during planning to increase efficiency. Contrary to our area of research, the initial state is assumed to be known and no exploration of the world is considered. Recently, Tenorth and Beetz presented KnowRob, a knowledge processing system for service robots [3]. KnowRob allows a robot to acquire and represent information from various sources, be it from the Internet or perceived by the robot's sensors. All sources are represented as virtual knowledge bases, queried during task planning, which employs deterministic and probabilistic reasoners.

System: The system is integrated using the Fawkes [15] and ROS [16] robot frameworks and middlewares. Bridge modules between the two let us incorporate components from both middlewares into our system and reuse prior work. In principle, our approach is independent of the particular robot platform. For now, we conduct our experiments using a PR2 robot, with the option to corroborate them on our custom-built robot Caesar [**?**] in the future.

III. Action Planning for Scene Observation

As robotic applications grow in complexity, integrating heterogeneous skills becomes a problem. Maintaining scripts

or state machines becomes difficult as the number of skills available to the robot increases. Symbolic planning offers one possible solution. Robot skills formulated as actions allow a domain-independent planner to compose a sequence of actions to accomplish a desired goal.

In household or workplace environments, the robot will be required to solve tasks such as finding and fetching specific objects. The challenge is to produce a plan to locate these objects in the environment. The objects might be located in cupboards or in a fridge in different rooms and the view might be obstructed. The robot must decide where to search for these objects. When objects are not completely visible the robot could move obstacles out of the way or move to observe the scene form another perspective. Another way to deal with occlusions could be picking up the object to bring it directly into the field of view of the sensors. In the current planning domain, the robot can move to a number of observation positions around a table. Each observation has the effect that new information is gained, without specifying what this new information will be. We formulate our planning goal so the robot will gather information by inspecting the scene on the table from all locations.

Despite recent advancements, only classical planning with deterministic closed world domains provides sufficient performance for high level control of a robot. Such planners usually cannot easily process geometric information. Consider a robot picking up an object from a table: it is difficult to express in symbolic logic if a kinematic solution for an arm motion to the object exists. However this information is crucial to produce executable plans. Therefore our planner Temporal Fast Downward/Modules (TFD/M) provides a mechanism to integrate external geometric reasoners into the planning process via *semantic attachments* [17]. For example, to determine the next position the planner queries a navigation module to estimate the path costs.

Furthermore, the robot has to deal with actions that can have unanticipated results. Our approach relies on continual planning [18]. TFD/M is wrapped in an execution, monitoring and re-planning loop. After each execution step if the current plan still achieves the goal we continue executing the plan, otherwise we re-plan. Consider the robot found an object it needs to grasp, but cannot reach it—re-planning is necessary. The additional knowledge is considered and the new plan lets the robot move to a position close to the object.

IV. Automated Data Recording to MongoDB

Today most mobile robots do not have the ability to recall recent events, or even just recent pieces of data. Some processing modules do keep a bounded buffer of observations, e.g. to track object positions. But it is typically not possible to reference arbitrary data seen recently by time or other criteria. A robot memory is required to store the data encountered to recall recent events or data, for instance to generate a combined view or to remember object positions. This memory needs to be bounded in storage capacity on the robot or in time to maintain relevancy of the data.

In previous work [4] we defined requirements for a robot storage system that can record data at run-time. The system must be able to store any and all data in real-time and provide powerful retrieval features to query specific data. This allows us to accomplish the (time- or storage-bounded) persistent memory, which is lacking in most robot architectures.

We continue to use the document-oriented, schema-less database *MongoDB* [5] for this purpose. The database groups key/value pairs into (possibly nested) documents and does neither require nor enforce predefined data schemas. In that it strongly differs from classical relational databases. This aspect provides us with the flexibility to generate direct mappings from data structures used for communication to database documents, allowing us to seamlessly record new data types as they are introduced and to sustain developer knowledge about data exchange (and now storage) formats.

Modern robot systems often follow a component-based paradigm, where functionality is encapsulated in modules. These exchange relevant data which is useful to another module via an inter-component communication middleware, for example following a blackboard approach like Fawkes or a publisher/subscriber model as ROS. The recording software taps into the middlewares and can record any and all data transmitted with only minimal configuration. MongoDB's virtual file system *GridFS* is used to store arbitrarily large files by splitting the data into suitably sized chunks. In our scenario the relevant data to record are point clouds, transforms, and images. Meta data for point clouds and images is stored in documents with references to the actual data kept in GridFS. All data is stored along with timestamps, which are used to query for data from specific time ranges.

V. Multi-Perspective Perception

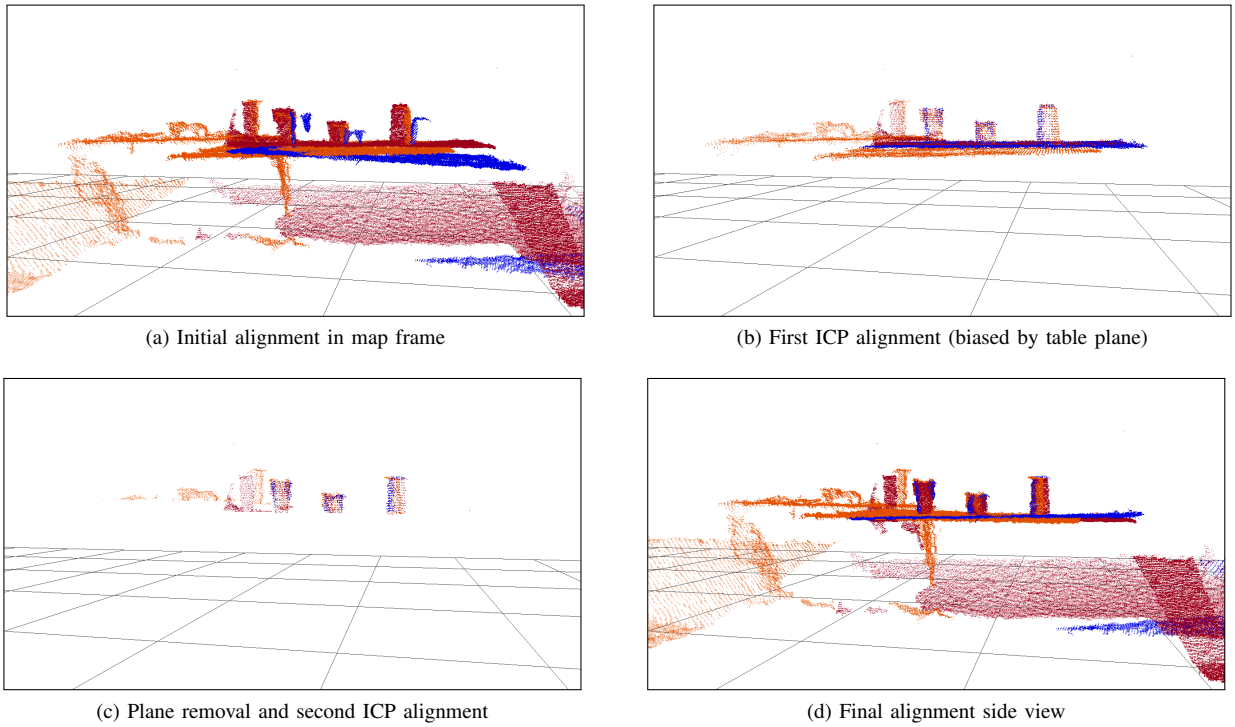
Occlusions in the perceived 3D point clouds often have a severe influence on the perception results. Objects of interest might be completely hidden, whereas partially-visible ones can be misclassified by the perception system. In many practical situations, the robot will not be able to cover the entire scene from a single perspective. Therefore, our goal is to combine point clouds from multiple distinct perspectives into one. We then apply our recognition pipeline on the aligned point clouds to identify the objects of interest. We now describe our perception pipeline in more detail.

Point Cloud Merging and Alignment

The task planner records timestamps of when the robot was at an observation position. When the planner invokes the point cloud merging pipeline it passes these timestamps as input parameters. The procedure consists of five principal steps. The most crucial steps of an example run of the described procedure are shown in Figure 2.

0. Restore Data Given the timestamps from the task planner, the point clouds closest in time are retrieved from the database. For a certain time window all transforms are restored into a temporary buffer to be used in the next step.

1. Initial Alignment To align the point clouds recorded from different locations around the table, our system uses the



(a) Initial alignment in map frame

(b) First ICP alignment (biased by table plane)

(c) Plane removal and second ICP alignment

(d) Final alignment side view

Fig. 2. Intermediate and final result images of the multi-perspective point cloud merging pipeline.

robot's pose at the corresponding time computed by Adaptive Monte Carlo Localization (AMCL) [19] in a given map of the environment. Together with the given pose of the sensor (in our case a Kinect camera mounted on the robot's head), this information is used to transform the point clouds from the sensor frame to a global reference frame (Figure 2a).

2. Filtering and Down-sampling Our system then filters the point clouds for outliers in terms of height over the ground support plane down-samples them to a voxel size of $1\,\mathrm{cm}^3$.

3. First ICP Alignment We then apply the iterative closest point algorithm (ICP) [20] to the point clouds to align them more closely. This alignment is particularly biased by the table plane in the different perspectives, as it typically accounts for most of the points (Figure 2b).

4. Plane Removal and Secondary Alignment In this step, dominant planes are detected and removed from the resulting point clouds. These are in particular the table plane and partial walls that happen to be within the filtering range. We then perform a second iteration of the ICP algorithm, which now favors the alignment of the actual objects on the table, as most other points have been removed (Figure 2c).

Figure 2d shows the final result of the point cloud merging and alignment procedure. In the following section, we describe the perception pipeline applied to the merging result.

Tabletop Scene Perception

In this step, we run our perception pipeline on the point cloud resulting from the merging and alignment procedure. In our experimental scenario, the robot is searching for specific cups of known dimensions on the table. Our system processes the point cloud to determine if the desired cups are on the table. This involves determining the most likely table plane and estimating its convex hull and normal vector. After eliminating all points corresponding to the table surface, the system computes the number of objects on the table through a clustering process. Each detected cluster is assumed to belong to one unique object. For each cluster (object), we compute a score representing its similarity to one of the cups the robot is looking for. This encompasses the similarity with respect to both the shape and size of the cup, where the shape is approximated by a cylinder.

The size similarity measure, sim_{size}, takes into account the three dimensions of the bounding box of the detected object. It is of the form $e^{-|d'-d|/\alpha}$, where d' is the measured dimension of the observed object, d is the true dimension of the cup, and α is a scaling parameter. The overall sim_{size} score is the product of the similarities with respect to all three (x, y, z) dimensions, resulting in a value between 0 and 1. To compute the shape similarity sim_{shape} of a cluster to a cup, we run a sample consensus segmentation process using a cylindrical model to fit the best cylinder to the cluster. We then compute sim_{shape} as the ratio of inliers (i.e. points lying on the computed cylinder or within a small distance from its surface) to the total number of points comprising the cluster.

The system then computes the overall similarity of a detected object to a specific cup by averaging both sim_{size} and sim_{shape}. Finally, each cluster is assigned the identity of the cup it resembles the most or is considered an *unknown object* if the overall score is less than a certain threshold θ.

VI. Experiments

In this section, we present an experimental evaluation of our system with its data recording, planning and perception

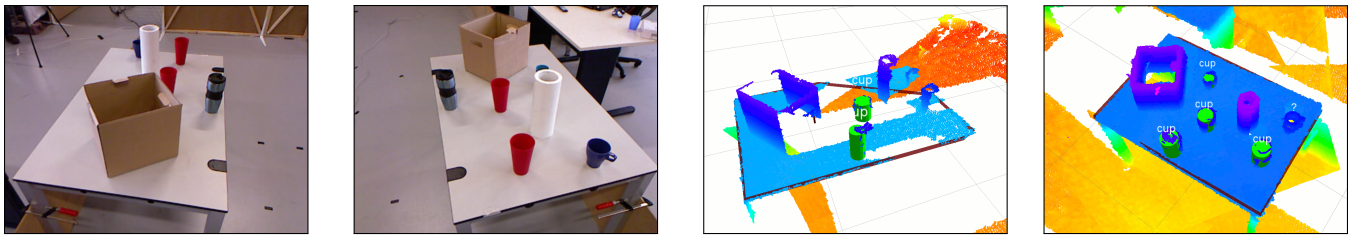


Fig. 3. The first two images respectively show the first and last perspectives of the robot's camera for the same scene. The third image shows the cup detection from the first perspective. Only two cups are recognized since one is occluded by the large box, whereas the other is too far and is only partially visible such that shape and size matching fail. The last image shows the proper detection after merging data from all five perspectives.

modules. We focus on how our multi-perspective perception approach improves the detection rates of objects in the scene compared to detection from a single point of view.

Experimental Setup

Our setup consisted of a table with several objects on it, which we want the robot to label either as cups or unknown objects (see Figure 1). As background knowledge, we provided the system with the dimensions of two different cups. When classifying the objects, the robot matched the different objects to either category according to their resemblance to those cups, as described in Section V. The robot can capture point clouds of the scene from five different locations, which were recorded *a priori*. We conducted four runs, each with a different arrangement of four cups and three or four other objects on the table. Objects like cardboard boxes were used to provide occlusions so that some object could not be observed from certain locations as shown in Figure 3.

System Performance

Three components are of particular relevance when it comes to the performance of our system. The *task planner* solves the problem at hand in less than 50 msec, which is negligible compared to the execution time for the robot movements. We expect the planning time to increase once we have more diversity in the required actions, e.g. once we employ manipulation for active perception.

The general *database recording* performance has been evaluated in-depth in [4]. Point clouds and images were stored regularly at 2 Hz at typical data rates of 12 MB/sec. Explicit periodic flushing was used to prevent data transfer congestion. Coordinate transforms in ROS and Fawkes are kept in a tree, where nodes represent reference frames and edges the transformation from the source (parent) to the target (child) frame. Each transform for an edge is published periodically. On the PR2, transforms are sent at typical rates of more than 3500 transforms per second. A single-host database can introduce undesirable latencies for this number of inserts. Therefore, we group transforms in intervals of 2 sec into a single document per transform link leading to insertion rates of only some 50 documents/sec, which the database could easily handle. With these improvements we achieved a steady data recording. The system scales with available storage bandwidth and capacity, which could be extended by using solid state disks or deleting old data.

The *point cloud merging* procedure has typical run-times of about 300 msec, including loading from the database and alignment. The *perception* pipeline is capable of operating at 30 Hz for single-view observations, and therefore easily at the speed of the merging step.

Perception System Accuracy

For each run, we recorded the number of correct object classifications made by the robot from each of the five perspectives. We compared the results for two setups. In the first, the robot ran the object recognition pipeline on the individual point clouds captured from the different perspectives independent of each other. In the second, the robot used the merged and aligned point clouds gathered from the different perspectives of the same scene and provided detection results after each merging operation. The results after each of the five perspectives are summarized in Figure 4 averaged over the four runs. The charts depict the precision and recall based on the number of true positives (correct labeling of a cup), true negatives (correct labeling of an unknown object), false positives (falsely labeling a object as a cup), and false negatives (labeling a cup as an unknown object). If a cup was completely undetected due to an occlusion, it was counted as a false negative.

Notice how the system based on single views of a scene is susceptible to many misclassifications compared to one which combines data from different perspectives. Partial point clouds captured of an object can easily result in wrong shape estimates compared to a merged point cloud capturing two or more sides of the object. The single-view setup therefore reported significantly more false positive cup detections leading to a low precision compared to our approach. On the other hand, our approach was more robust to false positives with a precision increasing in general with more added views. From some perspectives, our merging approach reported a false positive like labeling a mug with a handle as a cup, since it resembles the cups from most perspectives. This was typically corrected after adding a point cloud from a perspective that captures the handle as well, as the shape of the object is not consistent with a cylindrical model anymore. Moreover, we observed a consistent increase in recall after each added perspective of the table using our approach. In other words, the robot was able to detect all cups by the end of each run even though some were occluded at first. A video of an example run from our experiments can be found at `http://hybrid-reasoning.org/projects/c1`.

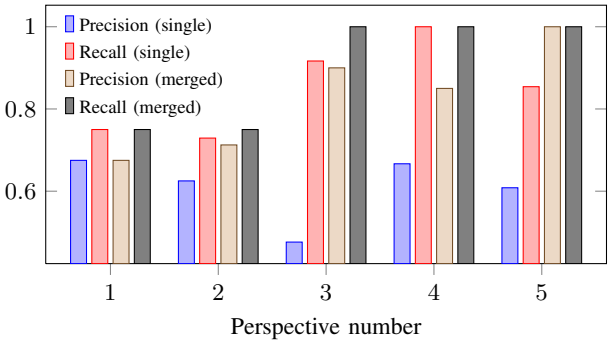


Fig. 4. Object classification results for two setups: using point clouds from individual perspectives only (single) versus using the merged point clouds from the consecutive perspectives of each scene (merged). Combining data from different perspectives resulted in better precision and recall at the end of each run (i.e. after going around the table). On the other hand, the precision of cup detection for the single setup was significantly more sensitive to the occlusions and shadows specific to the different perspectives.

VII. Conclusion and Outlook

Today, when developing a robot application, actions for knowledge acquisition are often explicitly encoded into the task description. In this paper, we have presented an architecture and building blocks of a system we intend to use as a baseline system for epistemic reasoning, i.e. if and how to gain the necessary information to accomplish its task.

So far we have developed a system that combines persistent memory and multi-perspective perception with continual planning to deal with (partial) object occlusions. We have points for further extensions that will allow us to easily implement and test new methods.

In the future we want to decide on the next observation position that promises the highest information gain by querying geometric planners using semantic attachments. A knowledge-based reasoner will explicitly represent the epistemic state combining qualitative and quantitative information to infer whether to acquire more information or to decrease uncertainty, and which actions to take to accomplish this. The persistent memory might cover more types of data to remember in the future, for example previously discovered object positions. Our current object recognition system already provides the necessary interfaces to plugin more sophisticated object detectors based on more comprehensive features, e.g. [21]. It also illustrates the value of combining data from different time steps to improve the quality of the acquired information. The multi-perspective merging could be made more efficient in the future by using OctoMap [22].

For the tabletop scenario, the multi-perspective approach has significantly improved our perception results and provides a capable baseline system with suitable extension points. We are currently developing a reasoning system that can automatically determine the information required to solve a task and an active perception system to acquire it.

Acknowledgments

N. Abdo, A. Hertle, and T. Niemueller were supported by the German National Science Foundation (DFG) research unit *FOR 1513* on Hybrid Reasoning for Intelligent Systems (`http://www.hybrid-reasoning.org`).

References

[1] M. Fox and D. Long, "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains," *Journal of Artificial Intelligence Research (JAIR)*, vol. 20, 2003.

[2] H. Levesque, R. Reiter, Y. Lésperance, F. Lin, and R. Scherl, "GOLOG: A Logic Programming Language for Dynamic Domains," *Journal of Logic Programming*, vol. 31, no. 1-3, 1997.

[3] M. Tenorth and M. Beetz, "KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. Part 1: The KnowRob System," *International Journal of Robotics Research (IJRR)*, 2013.

[4] T. Niemueller, G. Lakemeyer, and S. S. Srinivasa, "A Generic Robot Database and its Application in Fault Analysis and Performance Evaluation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[5] K. Chodorow and M. Dirolf, *MongoDB: The Definitive Guide*. O'Reilly, 2010.

[6] J. Mason and B. Marthi, "An Object-Based Semantic World Model for Long-Term Change Detection and Semantic Querying," in *IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2012.

[7] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE Int. Conference on Robotics and Automation (ICRA)*, 2011.

[8] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *10th IEEE Int. Symposium on Mixed and Augmented Reality*, 2011.

[9] D. Stampfer, M. Lutz, and C. Schlegel, "Information driven sensor placement for robust active object recognition based on multiple views," in *IEEE Int. Conference on Technologies for Practical Robot Applications (TePRA)*, 2012.

[10] M. Lutz, S. D., and C. Schlegel, "Probabilistic Object Recognition and Pose Estimation by Fusing Multiple Algorithms," in *IEEE Int. Conference on Robotics and Automation (ICRA)*, 2013.

[11] R. Eidenberger and J. Scharinger, "Active perception and scene modeling by planning with probabilistic 6d object poses," in *IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2010.

[12] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, and P. Jensfelt, "Search in the real world: Active visual object search based on spatial relations," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2818–2824.

[13] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical Task and Motion Planning in the Now," in *IEEE Int. Conference on Robotics and Automation (ICRA)*, 2011.

[14] J. Wolfe, B. Marthi, and S. Russel, "Combined Task and Motion Planning for Mobile Manipulation," in *20th International Conference on Automated Planning and Scheduling*, 2010.

[15] T. Niemueller, A. Ferrein, D. Beck, and G. Lakemeyer, "Design Principles of the Component-Based Robot Software Framework Fawkes," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2010.

[16] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[17] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic Attachments for Domain-Independent Planning Systems," in *Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2009.

[18] C. Dornhege and A. Hertle, "Integrated Symbolic Planning in the Tidyup-Robot Project," in *AAAI Spring Symposium - Designing Intelligent Robots: Reintegrating AI II*, 2013.

[19] D. Fox, "KLD-Sampling: Adaptive Particle Filters," in *Conf. on Neural Information Processing Systems (NIPS)*, 2001.

[20] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, 1992.

[21] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3D range scans taking into account object boundaries," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[22] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.