Inferring Generalized Pick-and-Place Tasks from Pointing Gestures

Nico Blodow, Zoltan-Csaba Marton, Dejan Pangercic, Thomas Rühr, Moritz Tenorth, Michael Beetz **
Intelligent Autonomous Systems, Technische Universität München
{blodow, marton, pangercic, ruehr, tenorth, beetz}@cs.tum.edu

***Abstract*—In this paper, we describe how to simplify the instruction of robots to perform pick-and-place tasks. This is achieved by inferring additional information about the environment, locations in it, and the used object, based on semantic knowledge. We use the Kinect RGB-D camera mounted on a PR2 robot and the human tracking system *OpenNI* that allows the interpretation of simple human gestures. Our application builds enables virtual movements of objects freely in space and the use of an ontology for reasoning about the plausibility of the tasks specified by releasing objects in certain locations. Our experiments show how relatively complex object manipulation tasks can be specified as simply as "drag-and-drop".**

I. Introduction

Specifying the tasks to be performed by a robot can be quite difficult, as common tasks such as placing an object into a container entail additional steps besides grasping, moving and placing. The robot has to locate the door of the container and its handle, find out how to open it, then open and close the door. On the other hand, many of these steps are common sense for humans. Just as path planning software is used to avoid the need for manual or step-by-step driving of the robot, semantic reasoning systems can be used to understand the human's instructions and generate the missing steps.

In this paper we present our first steps towards realizing such a system. We combine perception and interpretation into a system for robot task instruction through pointing gestures. This system detects and recognizes the objects pointed at, infers the type of the objects, and interprets the pick-and-place tasks implied by the pointing gestures through the use of the semantic environment model.

Inferring pick and place tasks specified through pointing gestures requires robots to infer the intended meaning. Objects can be virtually grasped, moved around and released by the operator in order to specify the task. If a human moves an object into a cupboard the intended meaning is to put the object inside the cupboard. If it gets moved on top of a table the intended meaning is typically to put the object on the table. As a consequence, the robot needs to infer the intended meaning by employing its background knowledge and in particular its semantic environment model as the associated knowledge base.

The *Semantic Object Map* used in our system contains rich information about each object's type, uses, etc. For example, a hierarchical representation of containers, doors, fixtures and opening trajectories enables the detection of the user's intent of placing an object in one of the containers and the generation of the required subtask of door opening and closing.

In order to aid the human operator, we present a visualization of the target object position, by hiding the original location of the object that was selected for the task, and virtually attaching it to the human's hand to intuitively move it to the desired 3D location.

Additionally, when verifying the task specification understood by the robot, spatial relations like "in" or "on" are more meaningful in the description than coordinates, and knowledge about the semantics and properties of objects (such as e.g. if object's content is perishable) aids in the detection of misinterpretations.

The main contributions of this paper are:

- the use of human tracking to guide segmentation of objects in RGB-D images;
- virtually hiding the original object location and possibility of moving objects around freely for easing the user's task;
- semantic reasoning based on the object's type and its required position in the environment for task description, verification and generation of pick-and-place subtasks.

While the number of semantic queries our system can answer so far is not too high, as the number and complexity of the possible task grow, so will the number of possible useful queries. The use of ontologies enables easy expansion in order to match the growing expectations from the system, and enables semantic perception by the robot.

We believe that the presented system provides an intuitive way to interact with a robot. Elderly people or patients who have to stay in bed (or similar situations where interacting with a PC is difficult and/or not natural enough for the user) could look at a screen and perform pointing gestures to control a robotic household assistant.

In the remainder of this paper we first give a short review of related approaches, we present the acquisition and content of our semantic environment map in Section III, followed by the description of human gesture interpretation and the virtual positioning of objects in Section IV and the semantic interpretation module which is described in Section V. We conclude in Section VI and present our agenda for future research.

** Graduate student authors in alphabetic order.

II. Related Work

The ability to use pointing gestures for directing robots has been presented in [8]. The authors use a laser pointer to "click" on 3D points, and select the appropriate action based on the coordinates, the robot state and surrounding context. We are taking a similar approach, but object selection and required placement positions are selected based on the human's pointing gestures. By allowing arbitrary target positions, not only points on surfaces, we can place objects also inside containers. This, combined with the use of the object's and environment's semantic interpretation enables us to infer more details about the required actions.

The semantic map of the environment is created based on our previous work [12, 13], and enhanced by the acquisition of opening trajectories for doors through interaction. The segmentation of drawers and furniture doors is the main advantage over alternative methods, enabled by the use of both 3D and RGB information.

Another approach for acquiring semantic maps describing basic elements like walls, floor, and doors is presented in [10]. A subsequent object detection step is presented as well. In [5] the authors extract windows, desks and entrance doors from point clouds by recovering alpha shapes and a Hough space classifier using the projection of rectangular structures. We already addressed the issue raised by the authors about building an ontology that allows for semantic reasoning [16], which we use here to derive additional required actions and to verify and semantically describe the placement.

The AI Goggles system [7] is capable of describing generic objects and retrieving past percepts of them using visual information. We are also using a vision based system for object recognition, that is open-source. We are able to recognize objects from an online grocery store[1] that we linked with the knowledge base to allow for reasoning on object properties.

III. Semantic Environment Map

In order to acquire the semantic environment map, our robot autonomously explores the environment and assembles point clouds containing spatial and registered visual information. We apply various segmentation methods in order to generate hypotheses for furniture drawers and doors. We extended our previous work presented in [13] to work on the less accurate sensing capabilities of the PR2, and make use of the robot's proprioceptive capabilities to create the final map.

The description of mapping is currently under review, but this is not the main focus of this paper, here we are using the generated map as resource that could also be provided manually by the user. However, we will present how we verify and improve the segmentation by the robot's interaction with the environment. This is the step needed to obtain the articulation models used during semantic interpretation in Section V. For the details about the whole semantic environment mapping process we kindly refer the reader to the following video: http://youtu.be/T15ycSmNOFY.

[1]www.germandeli.com

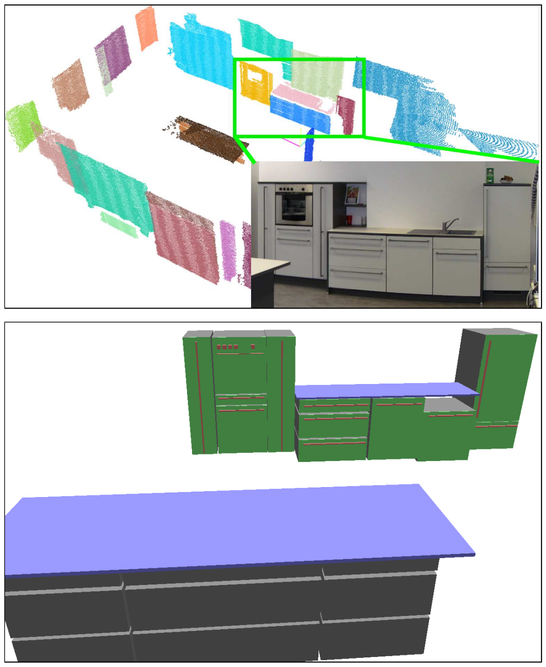
Fig. 1. Top: planar segmentation results (floor and ceiling not shown). Bottom: final map obtained by door detection through interaction with fixtures.

The idea of the robot interacting with the environment in order to overcome problems with uncertainties [4] or to verify grasp models for objects [3] has been present for a while. In this work, we describe a method to open detected drawers and cabinet doors without a-priori knowledge about the validity of the handle assumption or the underlying articulation model. In fact, we can estimate articulation models for every detected handle, which can be used in subsequent manipulation tasks directly with a more optimized strategy. Furthermore, we compute the regions of the point cloud which have changed after manipulating the handle and segment the differences, which gives us the correct segmentation of all furniture parts which are rigidly connected to the handle.

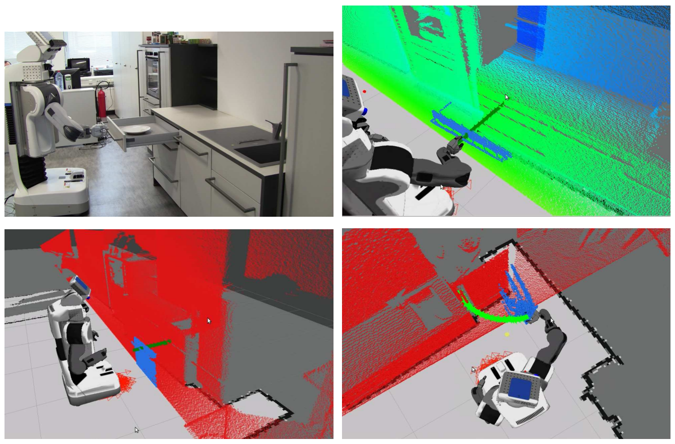
Fig. 2. Examples of interactive segmentation for two prismatic and one rotational joint.

A. *Opening of Drawers and Doors with Unknown Articulation Models*

We developed a general controller (see Algorithm 1) that makes use of the compliance of the PR2 robot's arms and the force sensitive finger tip sensors to open different types of containers without a priory knowledge of the articulation model. Since the arms lack force sensors, the algorithm uses the Cartesian error of the end effector (commanded vs. actual position) to determine when the maximum opening is reached. The algorithm relies on the grippers maintaining a strong grasp while the arms are compliant. This way the mechanism that is to be opened steers the arm in its trajectory even when there is a considerable difference between the pulling and the opening direction. The robot also adjusts its base position if the door mechanism requires this. The controller records a set of poses with the stable (aligned) grasps and returns those as an articulation model P. The controller works reliably as long as the force required to open the container is lower than the limit the friction of the gripper tips imposes.

Algorithm 1: Controller for opening containers with unknown articulation model. Note: poses are stored as transformation matrices (translation vector and rotation).

```
Initialize $p_0$ = point on the handle candidate;
$p_1 = p_0 + n_{furnitureplane}; t = 0$
while gripper_not_slipped_off AND cartesian_error < $threshold$ do
    if d($p_{t+1}$,projection of robot footprint) < .1 m then
        move_base(artif. workspace constr. for $p_{t+1}$)
    move_tool($p_{t+1}$)
    stabilize_grasp()
    $Rel = p_0^{-1} * p_{curr}$ with current tool pose $p_{curr}$
    Extrapolate: $Rel_s$ = scale $(Rel, (|Rel| + .05)/|Rel|)$
    $p_{t+2} = p_0 * Rel_s$
    t = t + 1
Return: Set of poses $P\{p_0 ... p_n\}$ representing the opening trajectory.
```

B. *Segmentation of Point Cloud Differences*

To segment out the front furniture faces we use temporal difference registration as put forth in [12], using a search radius parameter of 1 cm. We project the points that only appear in the second scan into the plane orthogonal to the last opening direction p_n. We obtain the convex hull in this plane, and assuming an environment based on rectangular furniture, we extract the width and height of the furniture front. For prismatic joints such as drawers, we can compute the distance between the two planes, which gives us a maximum opening distance and the depth of a drawer. For rotational joints, a similar value for the maximum opening angle can be found from the angles between the two planes. The depth of the container is in this case computed from the second cluster corresponding to the measurements of its inner volume. Results of this step for three furniture entities are depicted in Figure 2 and the final semantic map in bottom row of Figure 1.

IV. Specifying the Object and Target Position

In order to specify commands to the robot through gesture recognition and pointing we present a system that is based on the full body analysis framework *OpenNI* [1]. The latter takes input data from the Kinect sensor and provides a full body tracking approximated with fifteen degrees of freedom. The implementation and the video demonstrating the approach are available open-source on www.ros.org [2]. Our application layer on top of *OpenNI* consists of four nodes that allow virtual movement of objects of daily use from source to target destinations and are described in the following subsections.

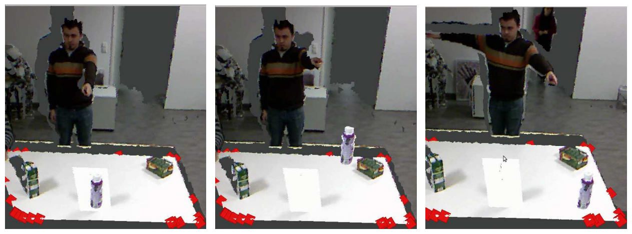

Fig. 3. Screenshots from the sequence of virtual object movement. From left to right: i) object grabbing, ii) object moving, iii) object release.

A. *Recognition of Gestures*

Since the data generated by *OpenNI* is rather noisy, we first applied a variant of the Simple Exponential Smoothing:

$$\boldsymbol{y}'_k = a y_k + (1 - a) y'_{k-1} \quad (1)$$

where a is the empirically determined smoothing constant and y_k and y'_k are the real and the smoothed joint angles respectively. In the next step we defined two gestures (left arm raised up and left arm stretched away from the body) in order to signal grabbing (start) and releasing (stop) commands. For the recognition of the actual gesture we compute the transform between the left hand joint and the neck joint and impose spatial, relative constraints for the transform to correspond to one of the gestures.

B. *Virtual Object Grabbing*

Once the start command has been published by the node above, this node subscribes to the point cloud data from Kinect sensor and segments out the object that the user is pointing at with the right arm. While the pointing direction is calculated from the transform between user's right arm elbow and wrist joints, the object is extracted by first fitting the horizontal plane to the point cloud using RANSAC-based approach and then extracting the point cloud cloud clusters within the polygonal prism of the found plane [14]. The first cluster that intersects with the line of pointing is then taken as an object to be moved. In the next step, its enclosing rectangle's corner pixels are

[2]http://www.ros.org/wiki/openni/Contests/ROS %203D/Teleop%20Kinect%20Cleanup

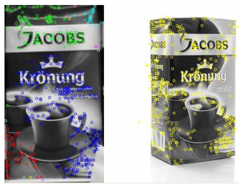
Fig. 4. Object recognition using vocabulary trees and SIFT features. Left: test image, right: matching template from www.germandeli.com database.

located and sent for filtering to the object filter node (next subsection) along with the plane equation and the RGB color extracted from points on the plane. Finally, the object cluster is transformed into the right arm wrist's coordinate system and re-published with every camera frame to reflect its new pose with respect to the user's arm (see Figure 3, middle).

C. Object Filter

This node also subscribes to the point cloud data from Kinect and it obtains a supporting plane, the two corner pixel coordinates of each object, and the plane fill color from the node above. These are all saved in a list, and whenever a new point cloud comes in, the pixels between the corners are projected onto the supporting plane (the positions could be pre-computed as well) and their color re-written with the color of the supporting plane. The processed point cloud is then re-published for the visualization (see Figure 3, middle).

D. Object Release and Recognition

The object release and recognition node remembers the object's pose upon receiving the stop gesture (shown in Figure 3, right), performs object recognition and sends both the pose and the object's semantic type to the knowledge base for semantic interpretation (see Section V). The recognition of the object uses RGB image of the extracted point cloud cluster and employs Scale Invariant Feature Transforms (SIFT) [6] using vocabulary trees [9] in order to robustly recognize textured objects (see Figure 4). The details of the architecture of this framework are currently under review, but the implementation is available open-source on http://www.ros.org/wiki/objects_of_daily_use_finder.

V. SEMANTIC INTERPRETATION

The semantic map is loaded into the knowledge base and represented as typed object instances. For details about the representation, see the KNOWROB-MAP paper [16]. The knowledge base further contains knowledge about these types of objects, for example that a refrigerator is the storage place for perishable goods, or that cupboards and the dishwasher are places where tableware can be found. Such knowledge is inherited by the object instances that are part of the map.

Furthermore, we connected the gesture and object recognition system to the knowledge base in a way that the target poses at which the objects are to be put down are automatically added to the knowledge base. The interface is realized similar to the K-CopMan system [11]. Using these methods, we can generate typed object instances at the target position the person is pointing at, and can use this information for further reasoning.

A. Semantically describing the target location

A first interesting step is to compute the relative location of the object: Is it inside a container, on top of a surface like a table or the counter top? Such qualitative spatial relations provide a more abstract and more semantic description of where objects are.

KNOWROB supports a variety of spatial relations that can largely be split into two groups: topological and directional relations. Directional relations, like *left of* or *behind* are relative to the position of an agent. Topological relations like *in* or *on*, in contrast, can be computed based on only the configuration of objects. In addition, we have the *connectedTo* relation in order to describe articulated objects and hinges. All relations are arranged in a hierarchy (Figure 5).

We use so-called *computables* [15] to calculate qualitative spatial relations based on the positions and orientations of objects. Computables are a kind of procedural attachment to the semantic relations that describe how these relations can be computed. This allows us to ask questions like "Where is object A?" in order to get all spatial relations to other objects. By querying for the *spatiallyRelated* relation, the system returns all sub-relations. Figure 6 visualizes the result of the following query that asks for a qualitative description of the location of *Cup67*:

```
?- rdfs_subproperty_of(Prop,knowrob:spatiallyRelated),
   rdf_triple(Prop,knowrob:'Cup67',Loc).

Loc = 'http://ias.cs.tum.edu/kb/knowrob.owl#Drawer13'
```

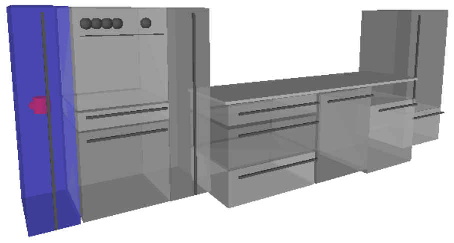
Fig. 6. Visualization of the query to the knowledge base for objects to which *Cup67* is *spatiallyRelated*. The result of the query is that *Cup67* (red) is inside *Drawer13* (blue).

B. Verifying object placement

The knowledge of typical storage locations of objects can be used to check if objects are to be placed correctly and alert the human if not. Common reasons for mis-placed objects are problems with interpreting the human motions, so double-checking the estimated poses for consistency with prior knowledge can help to improve the robustness of the system.

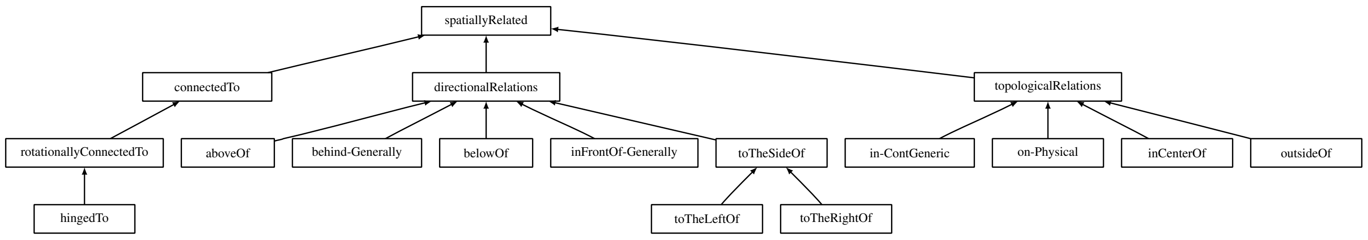

Fig. 5. Qualitative spatial relations

With the following query, we can ask if an object is located at a position that is defined as storage location for this kind of objects. The first line computes a qualitative description of the object's location, which is bound to the variable *Loc*; the second line then checks if the value of *Loc* is an appropriate storage location for the object *Cup67*. See Figure 7 for a visualization of the result.

```
?- rdf_triple(knowrob:'in-ContGeneric', knowrob:'Cup67', Loc),
   storagePlaceFor(Loc, knowrob:'Cup67').
```

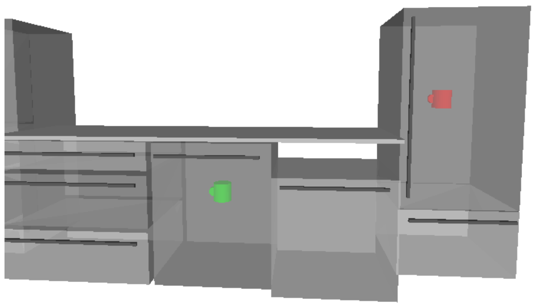

Fig. 7. Visualization of a query to verify if objects are placed correctly: Cups are supposed to be in the dishwasher, but not in the refrigerator, indicated by the green and red color.

C. Deriving additionally required actions

The computation of qualitative spatial relations provides the robot with a qualitative description of where to put the object. In some cases, this position is directly accessible, unless the location is inside a container. In these cases, the robot first has to open the container, resulting in additional actions that need to be taken. Our system supports to check if this is the case and, if yes, to read the trajectory that can be used for opening the container using the following query (whose result is visualized in Figure 8).

```
?- rdf_triple(knowrob:'in-ContGeneric', knowrob:'Cup67', B),
   rdf_has(B, knowrob:openingTrajectory, Traj),
   findall(P, rdf_has(Traj, knowrob:pointOnTrajectory, P),
           Points).
```

Remember that, during the mapping process, the robot opened all containers using a generic controller that can handle

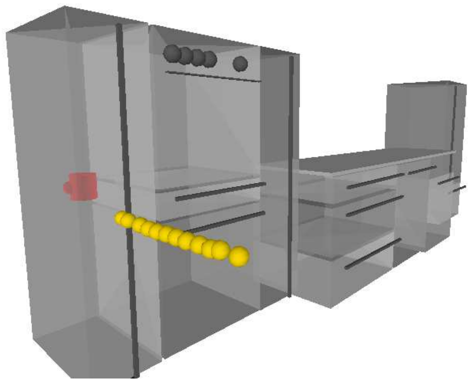

Fig. 8. Opening trajectory for *Drawer13* derived from the semantic map.

both rotational and prismatic joints. While opening them, the robot logged the end effector trajectory and attached this trajectory to the respective object instance in the knowledge base. The semantic description of a drawer, for example, looks like follows. The instance *Drawer13* of type *Drawer* is described by its dimensions, has links to its parts and to the instance of the map it is part of, and further links to an *openingTrajectory*. This trajectory is described in terms of the points it consists of.

```
Individual: Drawer13
  Types: Drawer
  Facts: widthOfObject 0.289
         depthOfObject 0.572
         heightOfObject 1.318
         describedInMap SemanticEnvironmentMap0
         properPhysicalParts Door16
         openingTrajectory ArmTrajectory1

Individual: ArmTrajectory13
  Types: ArmTrajectory
  Facts: pointOnTrajectory Point3D_13_1
         pointOnTrajectory Point3D_13_2
         pointOnTrajectory Point3D_13_3
         pointOnTrajectory Point3D_13_4
         [...]
```

VI. Conclusions and Future Research Issues

We have presented three possible ways of combining perception with semantic information in order to interpret human gestures, but of course as the considered tasks get more complicated the use of semantic information gets more important and extensive. Considering placements inside containers, it would be an advantage to infer the exact location of the placement. In the case of the fridge for example, the shelves, door or deep freeze compartments are both inside the container, but have distinct uses. The typical places for different object types can be encoded in the knowledge base, for example through interpretation of usage data.

While executing the plans by the robot falls outside of the scope of this paper, it is the next logical step. We plan to integrate our system into our high-level planning [2] in order to be able to consider other factors such as reachability, conflicting positions with other objects, etc.

Acknowledgments

We would like to thank our colleague Lucian Cosmin Goron for his valuable support. This work was supported by the CoTeSys (Cognition for Technical Systems) cluster of excellence, the EU FP7 project *RoboEarth* and by Willow Garage, Menlo Park, CA.

References

[1] Openni. 2010. URL http://www.openni.org/.

[2] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. In *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2010.

[3] Matei Ciocarlie, Kaijen Hsiao, E. Gil Jones, Sachin Chitta, Radu Bogdan Rusu, and Ioan A. Sucan. Towards reliable grasping and manipulation in household environments. In *Proceedings of RSS 2010 Workshop on Strategies and Evaluation for Mobile Manipulation in Household Environments*, 2010.

[4] Mehmet Dogar and Siddhartha Srinivasa. Push-grasping with dexterous hands: Mechanics and a method. In *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, October 2010.

[5] Markus Eich and Malgorzata Dabrowska. Semantic labeling: Classification of 3d entities based on spatial feature descriptors. In *Best Practice Algorithms in 3D Perception and Modeling for Mobile Manipualtion. IEEE International Conference on Robotics and Automation (ICRA-10), May 3, Anchorage, United States*, 5 2010.

[6] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. ISSN 0920-5691.

[7] Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi. Ai goggles: Real-time description and retrieval in the real world with online learning. *Computer and Robot Vision, Canadian Conference*, 0:184–191, 2009.

[8] Hai Nguyen, A. Jain, C. Anderson, and C.C. Kemp. A clickable world: Behavior selection through pointing and context for mobile manipulation. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 787 –793, 2008.

[9] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0.

[10] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Journal of Robotics and Autonomous Systems (JRAS), Special Issue on Semantic Knowledge in Robotics*, 56(11):915–926, 2008. ISSN 0921-8890.

[11] Dejan Pangercic, Moritz Tenorth, Dominik Jain, and Michael Beetz. Combining perception and knowledge processing for everyday manipulation. In *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, Taipei, Taiwan, October 18-22 2010.

[12] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3D Point Cloud Based Object Maps for Household Environments. *Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge)*, 2008.

[13] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Andreas Holzbach, and Michael Beetz. Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 11-15 2009.

[14] Radu Bogdan Rusu, Ioan Alexandru Sucan, Brian Gerkey, Sachin Chitta, Michael Beetz, and Lydia E. Kavraki. Real-time Perception-Guided Motion Planning for a Personal Robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4245–4252, St. Louis, MO, USA, October 11-15 2009.

[15] Moritz Tenorth and Michael Beetz. KnowRob — Knowledge Processing for Autonomous Personal Robots. In *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, pages 4261–4266, 2009.

[16] Moritz Tenorth, Lars Kunze, Dominik Jain, and Michael Beetz. Knowrob-map – knowledge-linked semantic object maps. In *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, USA, December 6-8 2010.