The International Journal of
Robotics Research
30(11) 1378–1402
© The Author(s) 2011
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364911415897
ijr.sagepub.com

Combined 2D–3D categorization and classification for multimodal perception systems

Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow and Michael Beetz

Abstract

In this article we describe an object perception system for autonomous robots performing everyday manipulation tasks in kitchen environments. The perception system gains its strengths by exploiting that the robots are to perform the same kinds of tasks with the same objects over and over again. It does so by learning the object representations necessary for the recognition and reconstruction in the context of pick-and-place tasks. The system employs a library of specialized perception routines that solve different, well-defined perceptual sub-tasks and can be combined into composite perceptual activities including the construction of an object model database, multimodal object classification, and object model reconstruction for grasping. We evaluate the effectiveness of our methods, and give examples of application scenarios using our personal robotic assistants acting in a human living environment.

Keywords

Perception system, multimodal classification, geometric categorization, 3D reconstruction, visual bag-of-words, online learning, modeling

1. Motivation

Autonomous robots that are to perform everyday manipulation tasks in human living environments must be able to pick up and place many objects of daily use. To manipulate them, the robots first have to perceive these objects. They are to detect the objects in realistic settings, categorize them, recognize object instances, estimate their 6D poses or even reconstruct their shapes from partial views.

Many algorithms have been developed to solve these problems for different subsets of objects, with varying accuracy and reliability, with different requirements for computational resources, and under different context conditions. Some of them require prior object models while others can do without, some infer only general categories, others exact instances without the knowledge of the broader categories these objects fall into. The approaches also differ in the type of sensors used, in speed, in that not all of them report 6D poses, in the number of objects they can deal with at once, etc.

The realization of robot perception systems that can perceive the range of objects to be manipulated in a typical human environment with the accuracy and reliability needed for grasping them successfully in real everyday settings poses a very hard and long-term research problem. Robot manipulation tasks are usually restricted to detecting distinctively textured objects, objects with distinct colors or specific shapes, based on the full list of possible objects.

Service robots, such as household robots, perform the same kinds of tasks with the same objects in the same environment over and over again. This enables them to learn and make use of more specific perception mechanisms for the particular objects and environments through *task and environment adaptation* (Horswill 1995). Task and environment adaptation thereby enables the robot to better perceive the objects in its environment by exploiting its experience and considering only relevant objects.

We consider that perceptions tasks can be intuitively categorized into *active* or *passive*. In the first type of task, a specific object or a set of objects needs to be located and thus the robot has to actively search for them. In the second case, the robot, possibly while performing other tasks, observes the environment or a specific region and identifies the different objects. For example, while setting the table for breakfast, the robot might spot a bottle of wine somewhere. Later, when the robot is to set a table for a lunch that includes a wine beverage, it does not need to actively

Intelligent Autonomous Systems Group, Technische Universität München, CoTeSys Central Robotics Laboratory II, Munich, Germany

Corresponding author:
Zoltan-Csaba Marton, Intelligent Autonomous Systems Group, Technische Universität München, Karlstraße 45, CoTeSys Central Robotics Laboratory II, Munich 80333, Germany
Email: marton@cs.tum.edu

start searching for a bottle, but instead retrieves the bottle's location from memory and navigates directly to it.

In both types of perception tasks, some sort of model is needed that describes what constitutes a certain object or an object of a certain type. Since we are most interested in pick-and-place scenarios such as completing a table setting (which encompasses both *active* and *passive* perception), models that aid the robot in grasping are of great importance. While identifying objects based on their appearance works reasonably well (Kragic and Vincze 2009), a 3D model is required in order to manipulate these objects and it has to be matched to the observed model. Also, as concluded by Kragic and Vincze (2009), a problem of current vision systems is robustness and scalability.

In this article we investigate object perception mechanisms for autonomous robots performing everyday manipulation tasks. The perception mechanisms use general perception methods to initialize learning, adapt and specialize their strategies as much as possible to the environment and objects it has to use. In this way, some of the problems become easier to deal with, thus improving efficiency.

The proposed object perception system maintains an *object model database* that provides information for the detection, recognition, pose estimation and grasping of the objects in the environment. We present a system that can use and build such a database under some assumptions and discuss the scalability issues. Our approach is to start with general detection routines and categorization modules to limit the possibilities for identities of objects and then add more and more details until an acceptable solution can be found. As the most important perception methods, texture-based and 3D perception routines constitute the focus of this article. However, simple additions such as considering physical size, color classification, and heat sensors can easily be added to the system's repertoire of examination methods.

The use of multiple detectors is important, as real-world objects, especially products of the same company have very similar appearance (coffee, cans, chocolate, etc.) as illustrated in Figure 1. Using only geometric descriptors is not generally applicable either, as, for example, mugs, drinks and cereal boxes typically have similar shapes but different appearance. In both cases, the intended use of the object can be significantly different and can be relevant to the task at hand. In addition, some objects might be hard to detect with one sensor, but more easily detectable using others (for example, semi-transparent or shiny objects), as shown in Section 7.4.

Although the set of objects of daily use that a personal robot could encounter in its tasks is unlimited, there are certain regularities that can be exploited with respect to the objects' shapes, textures and locations. Therefore, the perception system can adapt itself to a specific set of objects that are usually present in the world while at the same time retaining a certain degree of flexibility with respect to the incorporation of novel objects in its database of models. For example, a new flavor of iced tea should be recognized and manipulated as an instance of a box from its geometry, even though the robot has never seen it before. The knowledge about the iced tea can then be easily enriched with its appearance, provided it can be re-detected using geometric information.

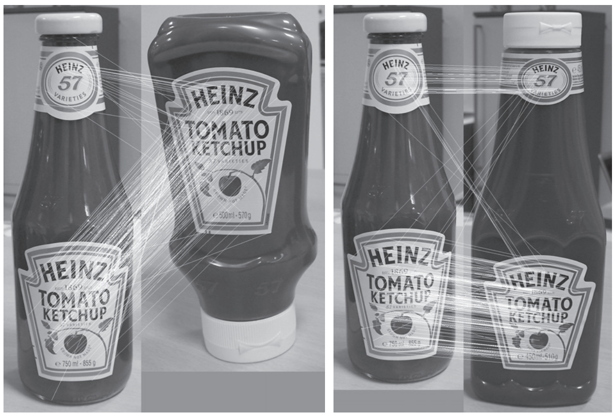


Fig. 1. Two examples of good model matches using SIFT features extracted from images. The images, however, depict three ketchup bottles with three different shapes which makes classification of the objects based on visual appearance challenging.

Also, certain information can be incorporated from external sources, for example a database of semantically labeled visual appearances of objects, which lacks the geometric appearance. In this case, the system should complete the missing information whenever possible. Similarly, in the case of autonomously learned object models, the semantic labeling or the correction of the assumed object identities could be done externally.

To this end, our system can learn the specific models of the objects it encounters. Since we do not assume that every perceived object in the environment is known, the system can mistakenly assume, for example, that a certain feature is unique to an object when in fact it is not. Our view is, however, that these mistakes are unavoidable during autonomous object learning; they are compensated for by the ability to automatically acquire models. Also, initializing the system with external databases (which are rather complete and labeled with semantically rich data, such as the KIT Object Models Web Database[1]) alleviates this problem significantly.

As an example, let us consider the case where the robot is to fetch a green object. Since this object happens to be the only green one it already knows about, it looks for the first green object it finds while heading towards the last observed position of the object. While it drives around, it observes various other objects that do not match the feature (color) it is looking for. However, since navigating around and taking laser scans takes longer than identifying the objects (scans with adequate resolution take of the order of seconds), the robot is constantly executing an 'identify-all-objects' routine. The latter updates the stored time-stamped position of recognized objects and assigns all unrecognized ones to new entries, assuming they are objects that were previously unknown to it.

As a simple, but effective way of considering different viewing angles, we implemented the following algorithms to aid the robot in integrating sequences of information. If, between subsequent scans, an unrecognized object candidate is found to have the same position as another object in the database, we inherit the class label from the object in the database. In case of, for example, incomplete models, this location-based identification enables us to incrementally learn the appearance of previously unknown views.

Mistakes in object detection and updates are possible of course, as in every classification problem, but the system can deal correctly with novel objects, something that is difficult for re-detection-based systems with a fixed set of objects. Attaching the semantic label to learned objects and the correction of mistaken object identities in the database is then performed by the human user.

We are building on, improving and combining several of our earlier works, mainly those presented in Marton et al. (2010a) and Marton et al. (2010b), and to some extent Blodow et al. (2009), Rusu et al. (2009a), Marton et al. (2009b) and Rusu et al. (2008b).

We adapted the model fitting and geometric classification to work on the less accurate 3D data coming from the eye-safe Hokuyo UTM-30LX. The number of geometric categories was reduced (small differences such as the existence of a handle are hard to detect with inaccurate sensors) and the results were made more robust by considering each view of each object as a separate class during training, but taking only the identified category into account during testing.

Visual classification was improved to be more scalable by a strategy similar to a visual bag-of-words from the computer vision field. The approach presented here is taken from the field of document retrieval, where word groups are used in a hierarchical tree setup to allow for fast lookup times. Please see Section 6.3 for more detail on the combination of a visual bag-of-words with document retrieval methods.

In brief, the main contributions of our this paper are:

- a perception system for object classification, categorization and modeling;
- automatic learning of descriptors for novel objects/views;
- a perception system based on affordable sensors, thus easy to rebuild.

The kinds of limitations of the system (that would require human supervision at some point) are as follows:

- previously unseen objects could be erroneously classified; this effect can, however, be minimized to the combined false classification rate of the detectors if the specialization step uses all of them;
- previously unseen faces of object could lead to the classification of a novel object if in the previous step it was not observed in the same location or identified correctly; this is unavoidable, but new views of known objects can be identified, given continuous observations;
- known objects can be misclassified for example in cases where objects have identically textured faces (some cereal boxes for example); this issue also occurs in re-detection-based systems, but if the object is continuously observed, and one of the ambiguous faces is not the first one that is seen, misclassification can be avoided.

Some of the above issues can be resolved by an object identity resolution system as we described in Blodow et al. (2010) (see also Section 8.3), which maintains a probabilistically modeled belief state of where certain objects are located given the sensor measurements, however, this in itself is a complex task and falls outside of the scope of this paper.

In the next section, we describe the architecture of our system, followed by the presentation of related work through the prism of different domains. Next, we give technical details on the overall implementation of our system and then describe the components of the different modules in Sections 5 and 6. We evaluate the different functions of our system in Section 7, and in Section 8 we show specific applications that benefit from our perception system. We conclude in Section 9 and present which next steps we planned for improving the system.

2. System overview

Figure 2 shows the schematic representation of our system, starting from the sensors of the robot used for localization and perception of the environment, through the possible *object detection methods*, and the selection of modules from the *examination method library*, to object recognition and learning using the information stored in the *object model database*.

Our two robots share similar setups, with standard cameras in stereo setups, and a tilting Hokuyo UTM-30LX for obtaining 3D point clouds. Lasers close to ground level provide information for localization. While many other sensors are placed on both robots, these are those most relevant to the presented system.

The images and 3D information coming from the robot's sensors are processed by the *perception executive* and the gathered data is interpreted according to the task at hand (searching for a specific object or identifying all objects). First, to limit the search space for object locations, a set of possible locations is extracted and the corresponding sensor readings (3D clusters, 2D regions of interest [ROIs]) are considered to represent object candidates. These object hypotheses are then processed as needed (see Figure 3 and Section 6) in order to associate the percepts to the correct object in the *object model database*.

When an object is being sought for, the system selects a set of features, whose values uniquely describe the object

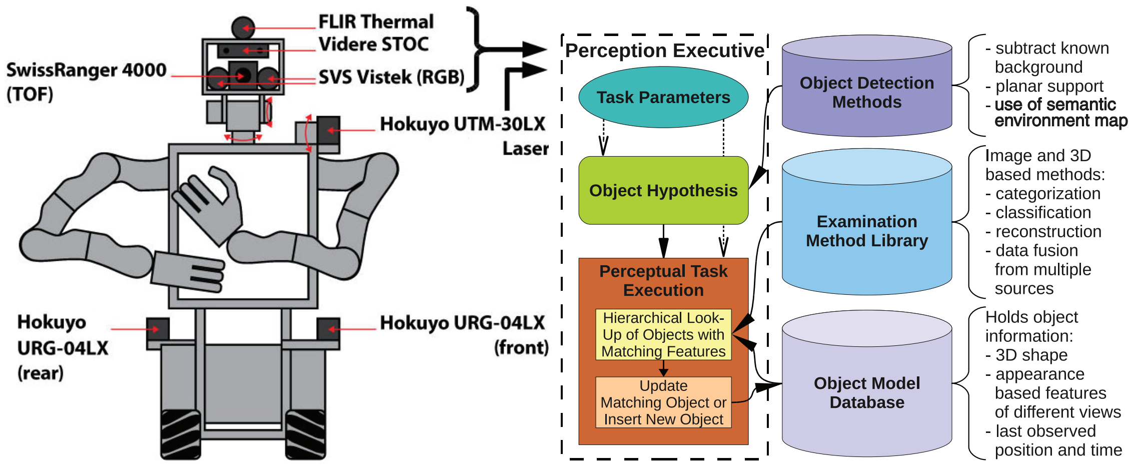

FLIR Thermal
Videre STOC
SVS Vistek (RGB)
SwissRanger 4000 (TOF)
Hokuyo UTM-30LX Laser
Hokuyo URG-04LX (rear)
Hokuyo URG-04LX (front)
Perception Executive
Task Parameters
Object Hypothesis
Perceptual Task Execution
Hierarchical Look-Up of Objects with Matching Features
Update Matching Object or Insert New Object
Object Detection Methods
Examination Method Library
Object Model Database
- subtract known background
- planar support
- use of semantic environment map
Image and 3D based methods:
- categorization
- classification
- reconstruction
- data fusion from multiple sources
Holds object information:
- 3D shape
- appearance based features of different views
- last observed position and time

Fig. 2. Object processing pipeline architecture: from sensor data to objects.

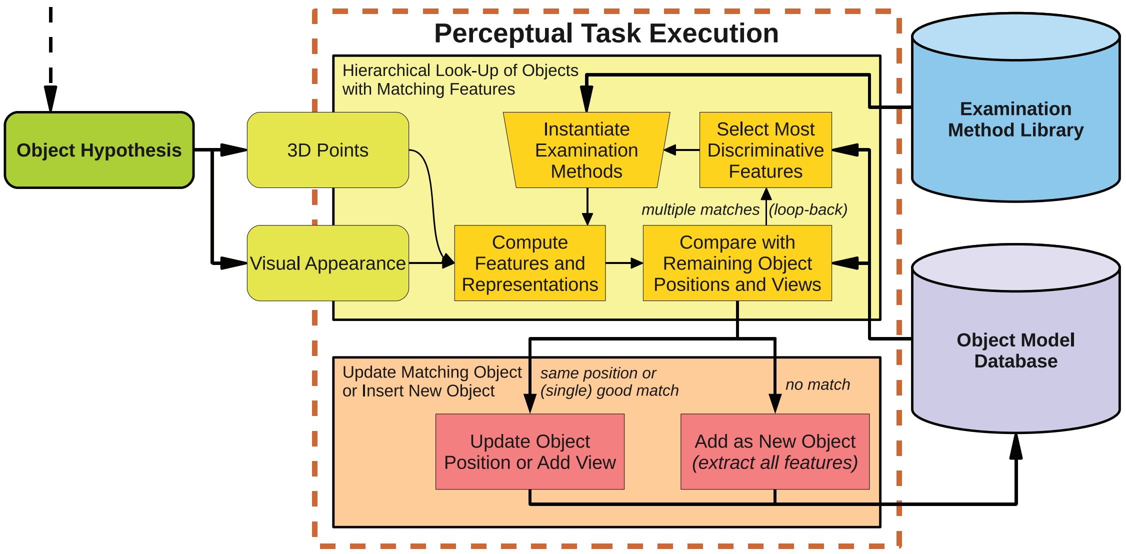

Perceptual Task Execution
Object Hypothesis
Hierarchical Look-Up of Objects with Matching Features
3D Points
Visual Appearance
Instantiate Examination Methods
Select Most Discriminative Features
multiple matches
(loop-back)
Compute Features and Representations
Compare with Remaining Object Positions and Views
Examination Method Library
Object Model Database
Update Matching Object or Insert New Object
same position or (single) good match
no match
Update Object Position or Add View
Add as New Object (extract all features)

Fig. 3. Details of the *perceptual task execution* module's operation. Selecting and combining methods from the *examination method library* to recognize known objects (either based on their stored features, or by co-locality in subsequent cycles), and to update their position or add their new view. The loop-back is not needed if a single object is being searched for. In cases when new objects were found, all features and representations are computed from the available data in order to be inserted into the *object model database*.

amongst all of the objects in the database. The same features' values are then computed using the examination modules for each object hypothesis, and the first one that presents matching ones is selected as the target object for, e.g., grasping. In the case that no geometric model is associated with the database object in question, we compute it on demand and feed it to the grasp planner.

If computational resources allow all object hypotheses can be checked against all objects in the database, and new objects, or new positions or views of known objects, can be detected. In this case, the features for each object hypothesis are computed one by one, according to a hierarchy, and the possible object identities are filtered in each step. The selection of the feature to be used in each step is hand encoded as of now, but an expansion of the single-object case is envisioned to be extended for finding the most discriminative features in each step.

This process is repeated until either an object is found whose stored features match all observed features, or until there are no matching objects left in the database, signaling that a novel object was observed. In the ambiguous case, when all features were computed and there are still multiple matching objects left from the database, the system takes no action and leaves the object hypothesis unclassified. Note that this is the case only if no visual features are present for the cluster (or stored for the possible matching objects), as visual feature classification assigns each observation to at most one modeled object, as discussed in Section 6.3.

Since the computation of the features for an object hypothesis is not too expensive, as shown in the corresponding sections, the aim of the procedure is to minimize the number of objects that have to be compared against as drastically as possible in each step, and allow a large number of objects to be handled efficiently. Given enough descriptive features, this method can scale well in the context of objects of daily use in human living environments. We consider this approach to be a move away from bottom-up, rigid pipelines, towards a more flexible setup. This enables the robot to specialize to the current situation, producing shorter processing times as not all of the methods are needed all of the time.

In order to be able to learn more and more about an object, multiple detections with different sensors and from different points of view are needed. To check if two percepts belong to the same object or not, we use a simplified version of the probabilistic framework for identity resolution (Blodow et al. 2010), which is based on positions of objects.

3. Related work

In the following sections, we discuss related approaches in groups representing the main areas of research this work falls into.

3.1. Perception systems

Nakayama et al. (2009) presented the AI Goggles system, which is a wearable system capable of describing generic objects in the environment and retrieving the memories of them using visual information in real time without any external computation resources. The system is also capable of learning new objects or scenes taught by users, however the training and testing phases are separate. As the core of the system, a high-accuracy and high-speed image annotation and retrieval method supporting online learning are considered. The authors use color higher-order local auto-correlation (Color-HLAC) features and the canonical correlation analysis (CCA) algorithm to learn the latent variables. In this work, we present a method for combining multiple features, and discuss how unknown objects can be detected and learned automatically.

Another system optimized for speed is MOPED (Torres et al. 2010). It builds on POSESEQ, a state-of-the-art object-recognition algorithm and demonstrates a massive improvement in scalability and latency without sacrificing robustness. The authors achieve this with both algorithmic (different variations of nearest-neighbor search) and architectural (single input multiple data [SIMD] instructions, etc.) improvements, with a novel feature-matching algorithm, a hybrid GPU/CPU architecture that exploits parallelism at all levels, and an optimized resource scheduler. Since the system considers visual-only features, it has certain limitations that can be tackled by considering 3D data as well. Our system uses a different visual classifier, but in theory any method can be used. The main advantage of our approach is in the combination of image and 3D features, and the building of models for grasping and re-detection for new objects.

A system for 3D perception and modeling is presented in Arbeiter et al. (2010), that can be used to reconstruct a 3D environment or learn models for object recognition on a mobile robot. Both color and time-of-flight cameras are used, and 2D features are extracted from color images and linked to 3D coordinates. Those then serve as input for a modified fastSLAM algorithm for rendering environment maps or object models. While the 3D aspects are not given that much importance, dealing with repeating visual landmarks on an object is handled nicely by the system. As the focus is on determining correspondences between scans, recognizing when an object is novel and has to be learned is not addressed.

A self-referenced 3D modeler is presented in Strobl et al. (2009), where the authors demonstrate that an ego-motion algorithm tracking natural, distinctive features and a concurrent 3D modeling of the scene is indeed possible. The use of stereo vision, an inertial measurement unit, and robust cost functions for pose estimation in the system further increase performance. While here the authors build accurate models using a hand-held device, we are using the robot's own sensors and movements to generate views that are similar to what the robot will encounter when it needs to perform recognition.

Incremental learning and recognition of objects is done in an unsupervised manner in Triebel et al. (2010), but the authors focus mainly on furniture pieces, and it is not clear how well multiple objects could be reliably detected without any prior information. Our approach is semi-supervised, where the robot generates object hypotheses on its own, but the categorization happens into user-defined classes.

A similar approach to ours is taken by Hinterstoisser et al. (2010), where a new view of an object is added to the model if it is so different from the stored objects as to cause the matching score to drop below the detection level. As we are combining the image data with 3D information, we effectively eliminate the problem of scale and have a better segmentation of the objects, and therefore require fewer views to be learned.

3.2. Model fitting

Multiple sensors were used for solving similar tasks, such as cameras (Coates et al. 2009; Ulrich et al. 2009), stereo cameras (Hillenbrand 2008; Fritz et al. 2009), 3D sensors (Steder et al. 2009), and also their combinations to speed up or improve results (Klank et al. 2009).

A vision-based grasping system which segments objects on a table and constructs triangular meshes for them is presented in Richtsfeld and Vincze (2008). While the presented method is general and works for many objects, it creates complicated models for certain objects, which could be simplified through the usage of geometric primitives. A simplification of the modeling problem is used in Grasp-It (Miller and Allen 2004), where combinations of geometric shape primitives such as spheres, cylinders, cones and boxes are used to model each object.

A computer vision and machine learning based method is used by Saxena et al. (2008) to train classifiers that can predict the grasping points in an image. This is then applied to images of unseen objects. To obtain 3D positions of grasping points, the authors use stereo cameras, but their approach works reliably only to the extent provided by the training data. Another issue is the segmentation of objects, since grasp points are provided with no information about what objects are in the scene and to which of them do the identified points correspond. Bone et al. (2008) used an accurate line laser and a camera to build models and identify grasping points for novel objects with very encouraging results. However, the system was tested only on two objects, thus its scalability is not clear.

Available models of complex objects are decomposed into superquadric parts in Biegelbauer and Vincze (2007) and Zhang et al. (2004), and these models are matched to a point cloud. This needs a database of models, and moreover, their decomposition into superquadric components, which is often difficult to obtain. A random sample consensus (RANSAC)-based approach for model decomposition is presented by Schnabel et al. (2007), where a set of 3D geometric primitives (planes, spheres, cylinders, cones and tori) are fit to noisy point clouds. Since the point clouds presented there are complete, the authors do not need to reconstruct the missing parts.

Thrun and Wegbreit (2005) described a method for detecting and verifying symmetries in point clouds obtained from a single viewpoint, and they project the existing points according to the detected symmetry to obtain the backside. However, using our methods we were able to reconstruct surfaces by approximating them with shape equations and thus generate a complete model, which can be meshed with the required density or sparseness according to the speed and accuracy requirements of our applications.

3.3. Object classification

There are two principal mainstream lines in the area of the object recognition related research: one aiming at recognition of objects in camera images, and one using 3D depth data acquired through range scanning devices. Combining both of them leads to a hybrid approach and our work falls into this category. Depending on the type of perception data, various different 2D (e.g. Lowe 2004) and 3D (e.g. Rusu et al. 2008a) distinctive local features have been developed. Taken individually, however, they are still insufficient to solve the full object recognition problem as both are prone to failure in situation where texture-less objects are present or depth data is too noisy or ambiguous. Therefore, different research initiatives have decided to combine sets of local features and cluster them together using different metrics (kernels) in order to be able to infer the global identifiers for objects.

Fergus et al. (2003) have proposed an unsupervised scale-invariant learning scheme, in order to detect objects on a wide range of images. Objects therein are modeled as flexible constellations of parts using a probabilistic representation for all significant aspects of the object. The work exploits the expectation–maximization algorithm in a maximum-likelihood setting. The method of Romea et al. (2009) estimates six-degree-of-freedom (6-DOF) object poses in cluttered scenes by matching local descriptors to stored models. Since the objects present in household environments are often texture-less, our approach constitutes an important advantage over the above proposed research initiatives, which fail to work in the absence of well-textured objects.

Another approach to obtain 3D information directly from camera images is to project computer-aided design (CAD) models from a database to the image and search for good matches in the edges domain, as in Ulrich et al. (2009) for example. While this is a more direct method, it is still dependent on a database of CAD models.

The work of Ruhnke et al. (2009) uses an iterative matching procedure to merge similar models in an unsupervised manner. However, it is unclear how well the proposed algorithm would generalize to unknown, novel objects. Lai and Fox (2009) perform outdoor laser scans classification combining manual labeling and data downloaded from the Internet in an effort to achieve what the authors call domain adaption. While their presented recall curves outperform others, the number of objects is relatively low and household objects are less distinct. Steder et al. (2009) authors investigate the extraction of GOODSAC point features and object recognition from range images that are in turn computed from point cloud data sets. These object models are, as in our case, created from real 3D data but processed using the work of Ruhnke et al. (2009).

The combination of depth information with camera images is addressed by Quigley et al. (2009). The authors calculate depth information for each pixel in the scene by applying laser-line triangulation with a rotating vertical laser and a camera. To obtain high-resolution 3D images, each scan requires 6 seconds with an additional 4 seconds spent on post-processing and triangulation. Thus, a waiting period of 10 seconds has to be expected before object detection and robot manipulation could be performed.

In the work of Xue et al. (2009) the grasping of objects modeled in the 3D object modeling center (KIT Object Models Web Database) was presented. The center employs a digitizer, a turntable and a pair of RGB cameras mounted to a rotating bracket which allows for views from above the scene. At present, there are around 110 highly detailed, high-precision objects publicly available. While working with such a system and data would yield high-quality results, its downside lies in the fact that the modeling center is rather expensive and cannot be used online, i.e. mounted on a mobile robot for autonomous mapping. In another initiative, the Columbia Grasp Database (Goldfeder et al. 2009) has been built. The major difference between this work and ours lies in how the models are obtained. The authors created artificial 3D models whereas we acquired our models by scanning real-world objects and surfaces, and are thus facing the problem of noisy and cluttered data.

4. Implementation details

The following sections detail the functioning principles of the building blocks of our system as shown in Figure 2.

Our system is developed within the ROS open-source framework (Robot Operating System[2]), as a collection of modules or 'nodes', that each tackle a specific sub-problem, and can either be run separately or as dynamically loadable libraries or 'plugins'. Thus, the data flow between nodes can either occur through shared memory, when optimal performance is required, or over the network, which is convenient for decentralized processing.

Using ROS and general examination methods based on PCL (Point Cloud Library[3])enables us to be platform independent in the sense that the system can be run on different robots, and results can be interchanged between them. So far, we have successfully performed our experiments on two different robots, TUM-Rosie (Blodow et al. 2010; Marton et al. 2010a; Pangercic et al. 2010) and the PR2 (Wyrobek et al. 2008).

Currently we employ the following methods to achieve the overall classification:

- auxiliary modules (noise removal, kd-tree, octree, described in the following);
- local feature estimation: Moving Least Squares (MLS) and Radius-based Surface Descriptor (RSD[4]) (Section 6.1);
- global (view-based) features: Global RSD (GRSD[5]) feature (Section 6.2), Scale-Invariant Feature Transform (SIFT) (Lowe 2004) feature using Vocabulary Trees[6] (Section 6.3);
- building geometric models: box/cylinder/rotational model fitting and triangulation (Section 6.4);
- categorization using support vector machines (SVMs) as described in Chang and Lin (2001) and classification using SIFT descriptors with vocabulary trees.

To increase the performance of the 3D data processing steps, we rely on noise removal (Rusu et al. 2008b), and efficient spatial decompositions such as kd-trees and octrees. Noise removal is performed to remove jump edges in the laser scans and to remove points from noisy or low-density areas of object clusters. We use kd-trees to find a fixed number of neighbors or all of the neighbors in a sphere with a given radius, while octrees are useful for down-sampling and occupancy grids, but also for the verification of the geometric models (Blodow et al. 2009), and for computing object-level features efficiently, as in Section 6.2.

5. Segmenting object hypotheses

Approaches to segment camera or point cloud percepts into distinct objects can be roughly split into two paradigms: Locating an instance of a known object in the scene, using, e.g., appearance models or feature descriptors, or segmenting the sensory data without prior knowledge about the objects or their views in order to acquire information about them.

In the first case, a direct search for each appearance can be performed, or a set of feature words can be extracted from the sensor data and matched to a model database as in Torres et al. (2010); Hinterstoisser et al. (2010). These approaches are usually limited by the fact that for locating one or more of a very large number of possible objects, performance can decrease significantly.

In the second case, as the objects to be detected are unknown, it is possible to segment scenes under certain assumptions, as presented in Figure 4:

- *Background subtraction:* Assuming successive addition of objects to the scene, one can easily detect successive changes even in cluttered scenes. More advanced methods include the learning of a foreground/background model.
- *Planar support:* Assuming a detectable support that assures physical stability allows for segmentation of objects in an Euclidean sense, as long as the objects are apart from one another (Rusu et al. 2009a). However, too much clutter is not acceptable in this case.
- *Environment maps:* Assuming a known 3D environment, such as stored in a semantic object map, makes it possible to define ROIs in which objects are expected, which can then be processed as in the *planar support* method. This approach is a natural extension to the 3D semantic mapping efforts.

The latter of these alternatives requires obviously the largest amount of knowledge about the environment. Since we have a static semantic environment map of our kitchen (Rusu et al. 2008b), we generated a URDF (unified robot description format[7]) map to describe the geometrical and kinetic properties of our environment. Every link in a URDF tree describes a part or reference frame in the map (e.g. walls, furniture items, appliances, doors, handles, etc.), and links are connected with joints of varying types, such as fixed for rigid links (such as those between walls and the floor), prismatic (such as drawers), or revolute (for rotational joints such as door hinges).

Applying labels to certain links, such as *counter_top*, *drawer*, *door*, *wall* and *shelf* lets us specify search filters that help in pre-segmenting the raw point data into meaningful regions, such as on top of certain counters, or the

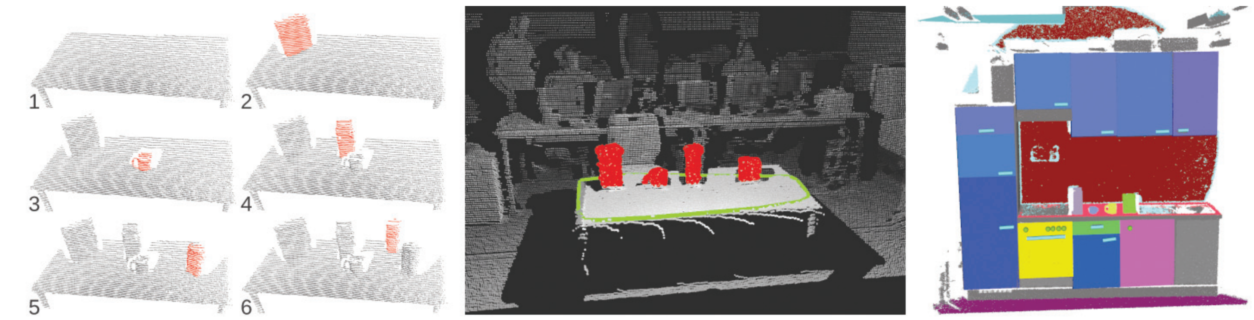


Fig. 4. Different object detection methods, from left to right: background subtraction, planar support, and using environment maps.

region in front of doors. The filters can reuse the geometric descriptions of the collision shape model stored in the URDF description and thus adapt the filter routines to each link label instance's distinct properties. These regions are processed by special routines that can, e.g., determine the opening angle of a cupboard door or the location of an extended drawer, or simply delete all points that represent walls to reduce further processing time. This results in faster processing times, since only some regions of the raw input data need to be examined, and every specialized processing step is executed solely on the task-relevant data.

For the scope of this publication, objects are located on supporting horizontal structures such as tables, shelves or other furniture, so we focus on object clusters found on them. However, for some of the training and test data acquisition, we fell back to the *planar support* method.

An interesting aspect of the tree-like structure of the map is the fact that for arbitrarily aligned rectangular or box-like search spaces, one can transform the input data such that the search space becomes axis aligned. We then identify complete subtrees in the map that share the same rotation and cache the transformed point cloud such that all search operations for these links can perform very fast axis-aligned bounding box tests with a minimal amount of point cloud transformations. A detailed analysis of this map-based pre-segmentation approach falls outside the scope of this paper and will be addressed in a separate publication.

6. Examination methods

In this section we present the examination methods used by our system, and their possible connections as summarized in Figure 5. Each method has a common interface, with initializing, processing, and result validating and returning functions, that enable the easy extension with additional methods if needed. Each method lists the data type it requires and provides, and the parameters are obtained through the parameter server of ROS. We are currently making efforts to enable the definition of processing pipelines using configuration files, but for now the steps are hard-coded.

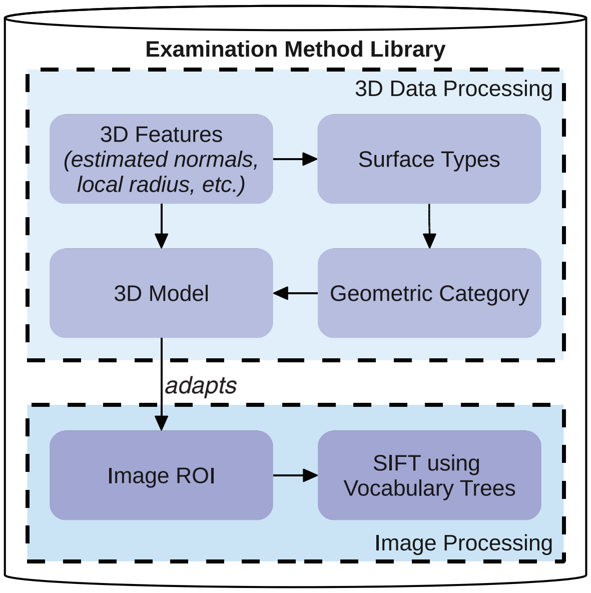

Examination Method Library
3D Data Processing
3D Features (estimated normals, local radius, etc.)
Surface Types
3D Model
Geometric Category
adapts
Image ROI
SIFT using Vocabulary Trees
Image Processing

Fig. 5. The conceptual representation of the complete pipeline that can be formed by the examination methods.

6.1. *Estimating descriptive local 3D features*

Since the point cloud coming from the 3D sensor is very noisy, we are using a moving least squares algorithm to smooth it. To speed up the process, and considering that small details are not detected by the laser which has errors in the centimeter range, we have simplified and thus sped up the Robust MLS method described in our previous work (Marton et al. 2009b), which was developed for a more accurate laser. To process the 3D points obtained by the Hokuyo laser, we use the principle component analysis (PCA) approximation of the tangent plane's normal, and re-fit the points and normals using MLS as shown in Figure 6.

While estimated surface normals are important, on their own they tell us little about the type of surface. The estimation of the curvature using the ratio of eigenvalues of a neighborhood's covariance matrix on the other hand

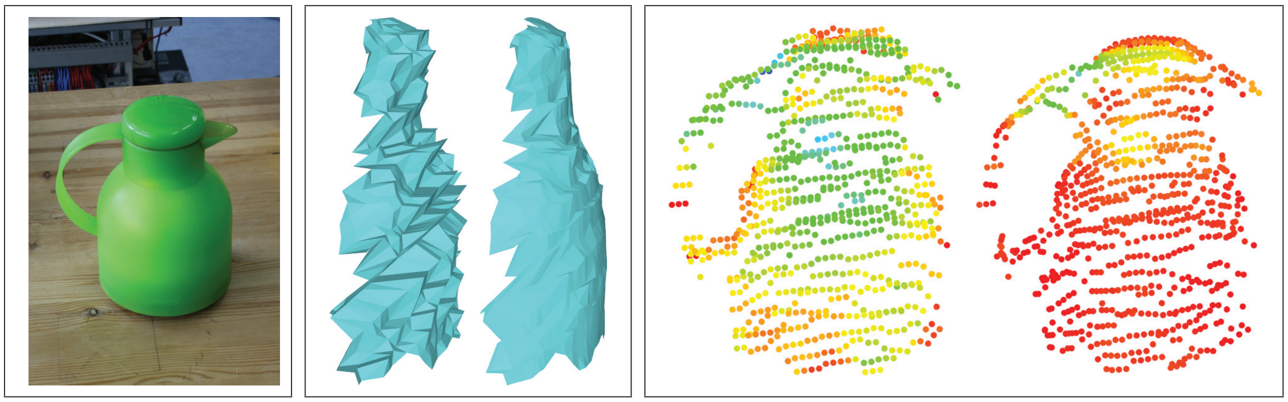


Fig. 6. Left: Photo of a teapot. Middle: Rendering from the side of the raw scan of the teapot obtained from the laser and the points projected onto the estimated underlying surface. Right: The estimated surface curvature values before and after MLS.

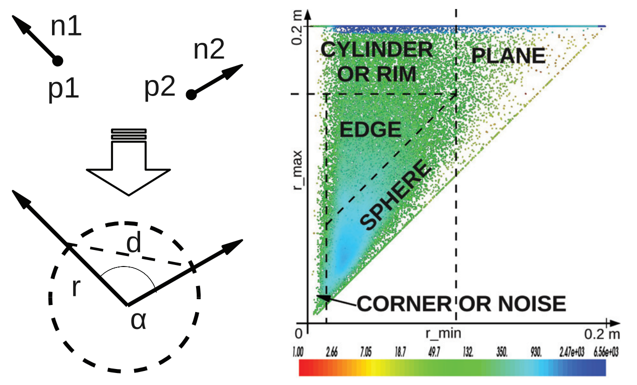


Fig. 7. Left: Illustration of how a distance and an angle between normals define a radius as per Equation (1). Right: Identifying same number of surface classes as using FPFH without classification. The plot shows the log density/number of points in a 1 cm range in the feature space of $r_{\min}$ and $r_{\max}$ for a variety of objects, with the rough ranges corresponding to certain shapes marked.

neglects surface normals and we found it to be too inaccurate. As argued by Marton et al. (2010a), the approximated radii of the smallest and biggest fitting curves to a local neighborhood are values with physical meaning, which can be tied directly to the underlying surface without the need for classification. We compute this feature (Radius-based Surface Descriptor [RSD]) starting similarly as in the case of spin images with a local support (Johnson 1997), with the added advantage that we consider the normals of the neighboring points directly, and that we can extract values that intuitively describe the local surface (as shown in Figure 7).

If we look at the case of the sphere, for each point all of the circles that fit to its neighborhood have the same radius, namely the radius r of the sphere itself. For each of these circles we can write the following relation between the distance d of a point on the sphere from the original point and the angle α between these two points' (undirected) normals:

$$d_{(\alpha)} = \sqrt{2}r\sqrt{1-\cos(\alpha)}. \tag{1}$$

From this we can see that given the distance and the angle between two point normals one can approximate the radius of the circle the point is on.

In the case of an ideal plane, this estimated radius will always be infinite with all neighbors, since they have parallel normals. A point on a cylinder is on multiple circles (ellipses actually), and the radius estimated with different neighbors will vary between the minimum radius (the radius of the cylinder) and the maximum radius (infinity). For corners and edges the estimated radius changes similarly as for spheres and cylinders (see Figure 8).

Given a point on a surface along with its neighbors, this minimum and maximum radius can be estimated using the model in Equation (1) by solving the equation system for r given the maximum and minimum angles for different distance intervals. To make the estimation easier, we can exploit the fact that[8] $\alpha \in [0, \pi/2]$ and the Taylor decomposition of Equation (1) is simple:

$$d_{(\alpha)} = r\alpha + r\alpha^3/24 + O(\alpha^5), \tag{2}$$

where $O(\alpha^5)$ indicates the existence of elements with order five and upwards. Thus, we can assume $d \approx r\alpha$ which renders the problem of finding r to a simple regression, as seen in Figure 9.

To find the minimum and maximum radius of a point based on its nearest neighbors, we compute the minimum and maximum angle in each of five distance bins, and compute the corresponding radius value. This introduces a small error ($r_{\min}$ is typically under- and $r_{\max}$ over-estimated), which we found to be less than 1 cm for all points of a set of 10 synthetic spheres with different surface radii R (for which $r_{\min} = r_{\min} = R$). For the real scan of the teapot in the right part of Figure 9 we found similar errors (its top cylinder having a radius of 4 cm and the bottom 7 cm).

In the cases where the two normals do not form an equal angle with the distance vector, the algorithm would assume they do and compute the radius in that manner. However, as the process is repeated for every point pair, and we are only interested in the minimum and maximum radius, these

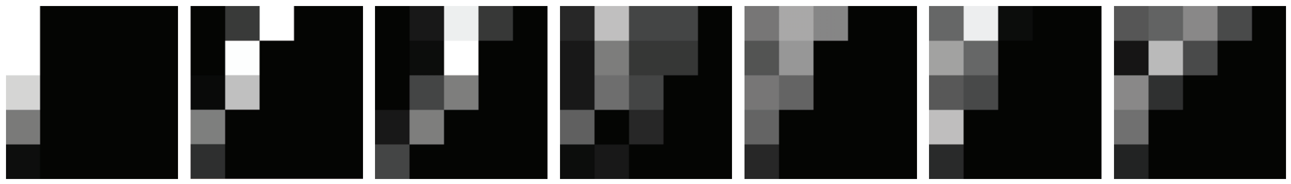


Fig. 8. Computation of the Radius-based Surface Descriptors (RSD): 2D histograms of normal angle difference and distance from a reference point to its neighbors (examples are shown from left to right for histograms computed for points on a plane, sphere, corner, edge and three cylinders, where the last two are computed for different points of the corrected teapot scan presented in Figure 6); from these histograms the physical surface radii can be read as the slopes of the different lines going through the lower left corner.

small errors are either evened out, or produce the typical result for curvature computation, i.e. estimated radii on a plane start dropping from very large to very small values the closer the point is to an edge. In the case of RSD this is also aided by the fact that the previous surface and normal estimation step also relies on a neighborhood of points. Please see Figure 14 for an example involving sharp edges (tea box).

This feature is easy to compute, while still being very descriptive (see Section 6.2) and does not require consistently oriented normals. Because it is a continuous value that estimates the real minimal metric radius of the curve each points lies on, it can be used for example as a prior when sampling points to fit different surfaces.

In our case, points with very high estimated minimal radius are preferred when fitting boxes, while they are avoided along with points having a very small radius when searching for rotational models. We incorporated this preference in the random sampling step for each model fitting algorithm. Since the surfaces scanned with these noisy sensors appear to be 'bumpy' even after MLS, the estimated minimal radius is in some cases smaller than it ideally should be. Still, they provide our RANSAC methods with a fast and accurate weighting for the points, depending on the model to be fit. The inliers are also weighted by how well their radius fits the model.

Based on the minimum and maximum radius it is possible to devise simple heuristic rules that categorize the surface types and enable the computation of a global descriptor, GRSD, as detailed in the next section.

6.2. *Object categorization*

We are using a two layered classification scheme for geometric categorization (Marton et al. 2010b), where local surface labels are combined in a global descriptor. To speed up the local surface classification, we label voxels directly instead of individual points, as those are needed for the global classification. This reduces the complexity proportionally to the average number of points in a voxel.

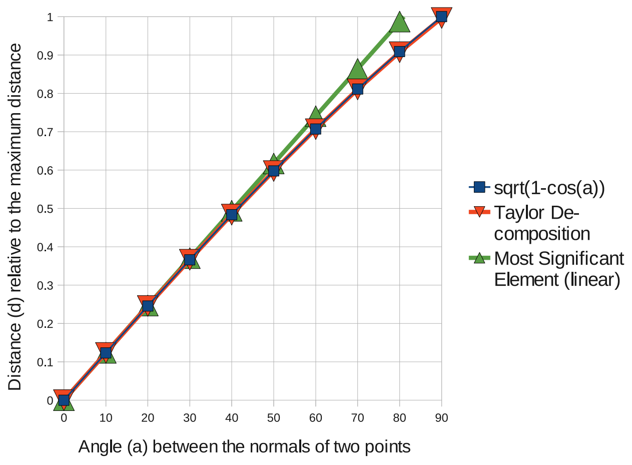
In each voxel with a width of 2.5 cm, we compute the minimum and maximum radius and label the surface by successive checks of the radii: planar ($r_{\min} > 0.1$), cylindrical (if not planar and $r_{\max} > 0.175$), sharp edge/corner or noisy (if not cylindrical and $r_{\min} < 0.015$), spherical (if not edge and $r_{\max} - r_{\min} < 0.05$) and rim (in the remaining cases). This is a simple and fast way to categorize surfaces (see Figure 7), and it divides the feature space formed by the radii well enough for the computation of a global feature. We picked the local surface labels to have an intuitive significance, but the actual correctness of these labels is not relevant for the global classification.

Once all voxels are annotated locally using a geometric class, our processing pipeline constructs a global feature space that can produce a unique signature for each object cluster. This space is based on the idea that, for a set of labeled voxels, a global feature can be constructed by observing the relationships between all these local labels (and the encompassed free space). Since the labels represent geometric classes obtained from the classification of RSD descriptors, we call this new feature the GRSD.

Algorithm 1: GRSD computation.

```
$L = \{l_1 ... l_m\}$ /* octree leaves encapsulating the
    object                                              */
$C = \{c_{l_1} ... c_{l_m}\}$ /* the leaf classes generated by
    RSD (e.g. class empty, class plane, etc.)
    */
$\mathcal{H}$ = ZeroMatrix /* transition matrix holding
    the number of neighboring classes                   */
foreach $l_i \in L$ do
    foreach $l_j \in L$ do
        $r_{ij} \leftarrow (l_i, l_j)$ /* create a line segment
            between $l_i$ and $l_j$                       */
        $I = \{l_k | l_k = L \cap r_{ij}\}$ /* get leaves
            intersected by $r_{ij}$                       */
        /* count the class changes between
            leaves in $I$                                 */
        foreach $l_k \in I$ do
            $\mathcal{H}[c_{l_k}, c_{l_{k+1}}]++$ /* increase
                counter                                   */
```

The complete algorithm is described in Algorithm 1, and Figure 10 shows two sets of feature vectors for different objects generated by the GRSD estimation. We were



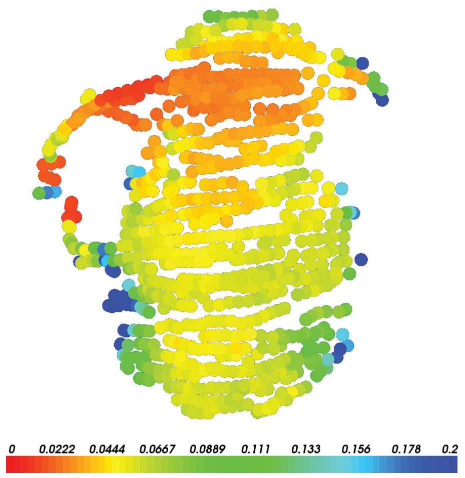


Fig. 9. Left: Variation of point-to-point distance by angle, measured from the center point of a half-sphere. Right: Radius estimation results for a scan of a teapot; values are color-coded from red (radius of 0) to blue (radius of 0.2 m or higher). Note that for visualization purposes the values were capped at 0.2 m, as planar regions have an infinite or very large radius. For the accurate color information we kindly direct the reader to the electronic version of the article.

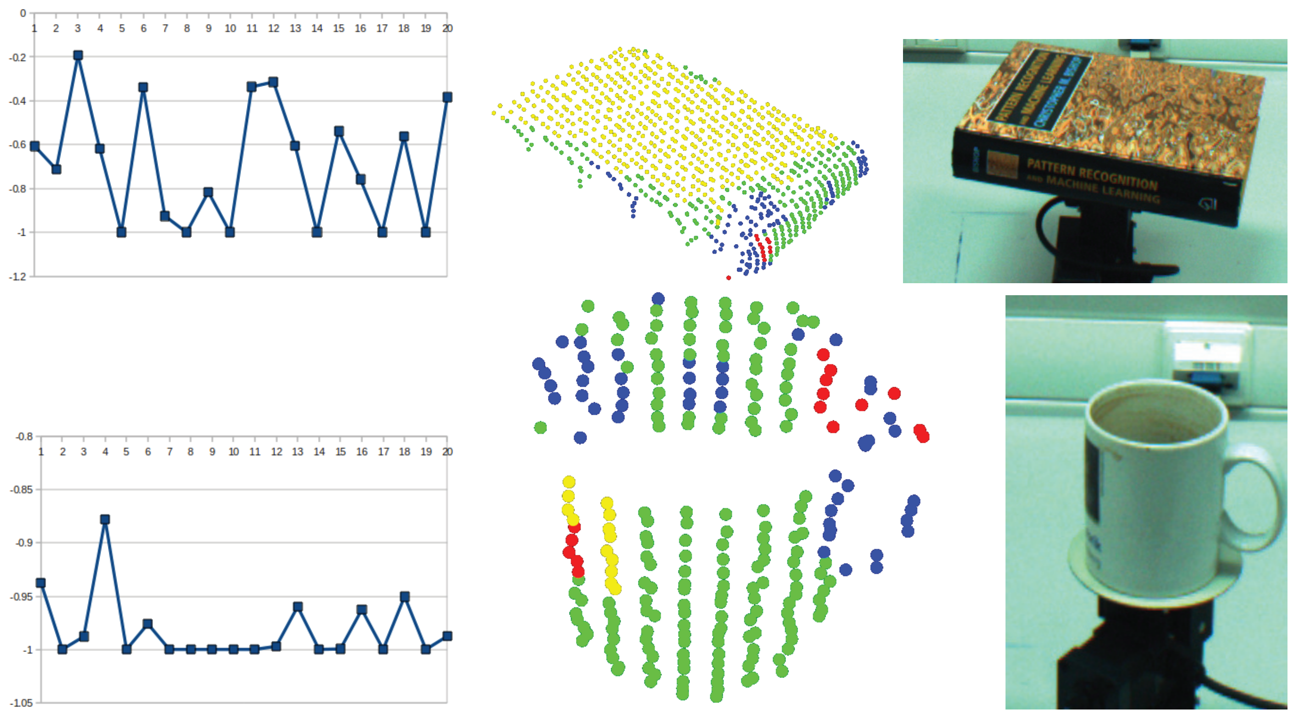


Fig. 10. Plots of GRSD histograms (left) and identified surface types based on RSD (middle) for a flat box (i.e. book, upper row) and a cylinder (i.e. mug, bottom row). Left: The GRSD histogram bin values are scaled between −1 and 1 (based on the value ranges found in the training data). Middle: The colors represent the following local surfaces: red, sharp edge/corner (or noise); yellow, plane; green, cylinder; light blue, sphere (not present); and dark blue, rim (i.e. boundary, transition between surfaces). For accurate color information we direct the reader to the online version of the article.

using the Octomap library (Wurm et al. 2010) to facilitate ray intersection tests in the voxels encompassing the objects.

The computation of GRSD is similar to GFPFH (Rusu et al. 2009b), but we use voxel-based labeling based on RSD, and we sum up the individual $\mathcal{H}$ histograms instead of computing their distribution, to further reduce computational complexity. In this way, the complete processing of a cluster (correcting, estimating normals, computing the voxelized RSD values, labeling voxels and constructing the GRSD) takes between 0.2 and 0.5 seconds (depending on object size) on a single core working at 2 GHz.

6.2.1. Selection of geometric categories The geometric categories were selected based on an analysis of the following databases containing objects of daily use: TUM Organizational Principles Database (**TUM-OPD**[9]), A List of Household Objects for Robotic Retrieval Prioritized by People with ALS (**ALS**) (Choi et al. 2009), KIT Object Models Web Database (**KIT**), Semantic 3D Database (**SemanticDB**[10]) and Household Objects Database from Willow Garage (**WG**[11]).

We selected six geometric categories, covering most of the objects a robot could perceive and manipulate in a kitchen, namely *mug/can sized cylinder*, *flat box*, *pressed-top box (Tetrapak)*, *box*, *tube-like cylinder* and *rotationally symmetric*. The statistics done over the different databases (Table 1) reveal a comparably small number of objects (such as scissors, money, sunscreen) that do not fall into one of the above categories (154/621, i.e. just under 25%). We included the *other* category into our geometric classification, and merged rotationally symmetric objects such as teapots, bottles, etc. into it. Since we group each view separately during training, adding very different objects to the *other* category does not make the category more ambiguous for the classifier.

6.2.2. Evaluation We trained each view of objects from the categories listed above as a different class using an SVM classifier. We used 934 individual views for training, at least 2 objects of each type scanned from different angles. For the final classification only the general category was considered, to avoid mistakes due to similar views, thus the model's accuracy for 174 newly segmented scans of the training objects (i.e. test views) was 93.68%, and we tested it using 205 views of untrained objects and had a success rate of 85.37%. The corresponding confusion matrices in Figure 11 show that we obtained good results for the distinctive categories, and had an around 30% confusion rate between two of the categories (between the side views of flat boxes and relatively thin Tetrapaks). The existence of similar views between differently shaped objects also makes the automatic learning of geometric categories quite difficult, producing analogous results to Figure 1, further supporting the need for combining perceptual mechanisms.

6.3. Visual feature classification

The visual feature detection is carried out only on the ROI in the image, in order to avoid false-positive matches with the background. To obtain the ROI the robot simultaneously takes a 3D scan and captures an image of the scene in front of it. The robot generates object hypotheses by detecting candidate point clusters in the 3D point cloud acquired by the depth sensor. These object hypotheses are then back-projected into the captured image as ROIs which are subjects used for detecting and recognizing objects.

For the SIFT-based detection we first determine segments by performing region growing on detected features in image space which typically results in an over-segmentation of the ROI. Identification of the object and the image region it belongs to is then performed through methods transferred from document retrieval. In document retrieval tasks for example as in Web search, we look for the documents that best match a given query term. To do so the search engines compute frequency statistics for discriminative words and word stems as a pre-processing step performed on all documents. Given a search term, fast indexing mechanisms quickly search for the documents that are particularly relevant with respect to frequency for the search term. The application of text retrieval technology for object matching is promising because it is very mature and the techniques aim at having high recall and precision rates while at the same time being very fast.

The computational idea of textual document retrieval can be mapped to object description matching in the following way. The descriptors computed from the ROIs that belong presumably to (or are partial views of) the same object are considered to be the search term. The object descriptors of the different views of the relevant objects are the documents of the document retrieval model. Word frequencies are replaced by the frequency of visual object features. Given a large set of objects, represented by their object descriptors, and the feature descriptor of an image region we can then index the objects where the particular features are particularly prominent using the respective methods from document retrieval.

In this paper we apply vocabulary trees for weighted term frequency inverse document frequency (TF-IDF) indexing (Robertson 2004), a method used in document retrieval to find the documents that best fit a given textual user query. In this reformulation of object identification, vocabulary trees speed up the retrieval of the matching objects.

The methods for object descriptor matching do not only match a given region descriptor to the large set of object descriptors, they can also learn new object descriptors to be put into the visual object library (see Figure 12).

6.3.1. Vocabulary tree The vocabulary tree approach to object classification is based on the bag-of-words (Sivic et al. 2005) document retrieval methods, that represent the subject of a document by the frequency in which certain words appear in the text. This technique has been adapted to visual object classification substituting the words with local descriptors such as SIFT computed on image features. The vocabulary tree of branching factor K and depth L is a tree data structure where the nodes in the tree represent a set of SIFT descriptors. The root node of the vocabulary tree represents the SIFT descriptors of all views of all object models in the library. If a node n in the vocabulary tree represents the set of SIFT descriptors $\mathcal{N}$, then its children nodes represent the partitioning of $\mathcal{N}$ into k subsets represented by the children nodes $cn_1, \ldots cn_k$ where the SIFT descriptors within a children node are similar and those of different children nodes are dissimilar.

Table 1. Frequencies of the selected categories of objects in different known databases of objects of daily use, based on hand-labeling through visual inspection.

Category	TUM-OPD	ALS	KIT	SemanticDB	WG
Mug/can sized cylinder	93	3	26	9	68
Plat box	23	5	28	4	3
Pressed-top box (Tetrapak)	4	0	1	4	1
Box	42	4	23	4	2
Tube-like cylinder	12	1	6	3	18
Rotationally symmetric	46	2	11	0	21
Other	99	10	15	11	19

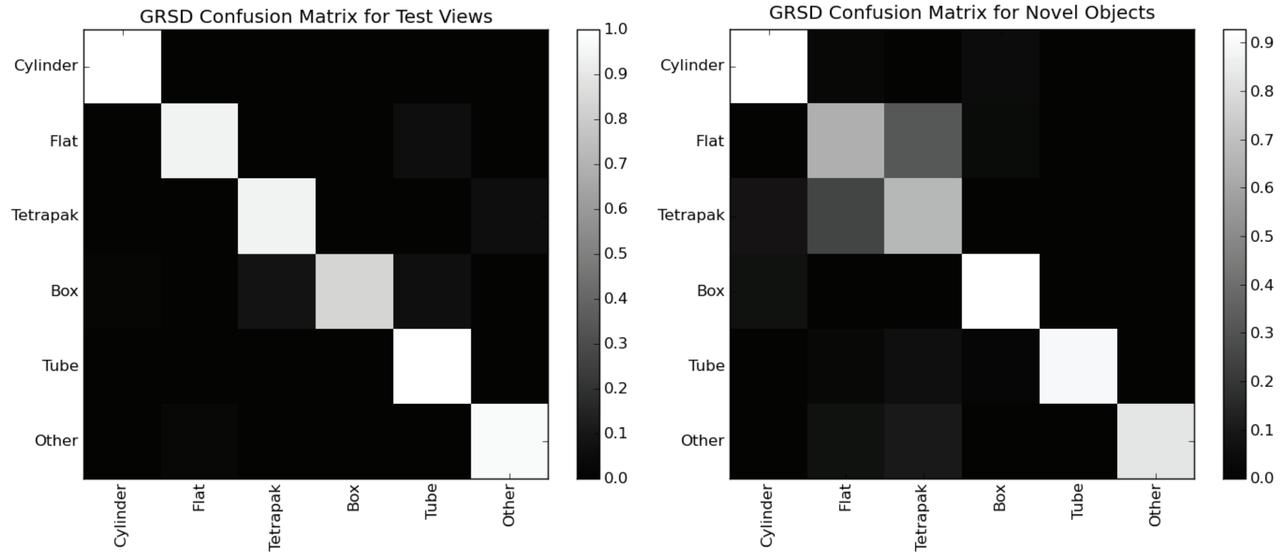


Fig. 11. Confusion matrices for new scans of the trained objects (left) and scans of new objects (right) provide the true and false-positive normalized statistics for the geometric classes (neglecting the view classification results) as discussed in 6.2.2. Ground truth is shown in rows, and classification results in columns. Please note the different scales in the two plots.

Thus, by taking a SIFT descriptor *sd* and classifying it hierarchically through the vocabulary tree using the defined distance measure on the SIFT descriptors we quickly find the set of the most similar SIFT descriptors in the *object model database* as the leaf nodes whose representative SIFT descriptors have the shortest distances to *sd*. For efficiency *sd* is not compared with all features in a given node, but to the centroid of its features.

The SIFT descriptors in the vocabulary tree also have a reference to the object model they occur in. Thus, when *sd* matches a leaf node it votes for the object models that the SIFT descriptors of the identified leaf belong to.

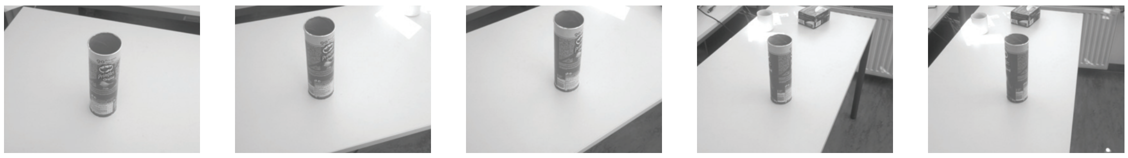
The children nodes $cn_1, \ldots, cn_k$ of $\mathcal{N}$ are computed by applying k-means clustering to the SIFT descriptors of node n. Since the weighted TF-IDF algorithm works on words (equivalent of leaf nodes), we use a vocabulary tree to convert the keypoint descriptors into words, where each word is an integer value corresponding to the number of the leaf node.

Building the database

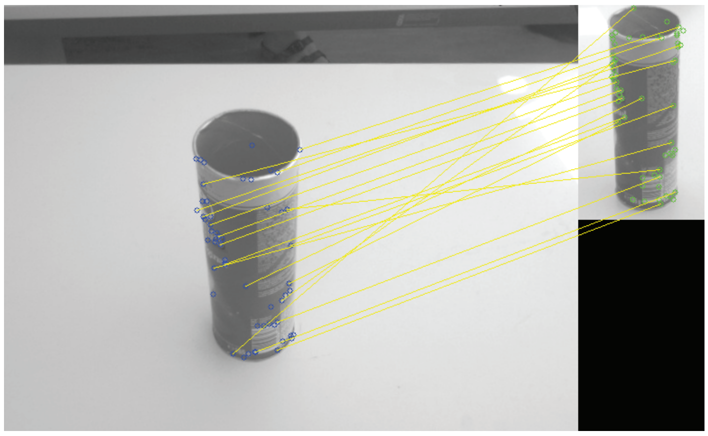
In our approach we use a similar database (*object model database*) to that described by Nister and Stewenius (2006). In order to be able to detect objects the database stores only the quantized SIFT features of the images, but not the images themselves.

Extracting SIFT features

In order to extract the visual SIFT features from the images we use an open-source implementation of the standard SIFT algorithm (Fast SIFT Image Features Library[12]) as initially described by Lowe (2004). Each SIFT feature is characterized by a 128-dimensional descriptor vector, 2 image coordinates, a scale and an orientation value. In the current implementation we use only the descriptor vectors for the detection process and the image coordinates for visualization.



(a) View 1 (b) View 2 (c) View 3 (d) View 4 (e) View 5



(f) Successful match against an online learned appearance model

Fig. 12. Top row: Different robot views used for online learning of SIFT appearance models. Bottom row: Successful match against an online learned appearance models.

Generating database documents

After we have the vocabulary tree, we quantize feature descriptors to single words. For every image, we take all SIFT features, we quantize them with the vocabulary tree and we group the resulting words in one document for every image. In this way the document is just the list of all quantized features corresponding to a single image.

Populating and training the database

After we have all image documents, we insert them into a specialized database as proposed by Nister and Stewenius (2006). The database is then trained with the weighted TF-IDF algorithm. After that it can be queried with documents generated from the input camera images in order to find the best database matches between objects in the image and the objects in the database. The database documents along with specific database information can be stored in a binary format in order to allow for fast loading of the database. Additional information like image file names, textures and feature coordinates are also saved for the visualization purposes.

The whole detection process is implemented as a single ROS node, which receives an image coming from the camera and outputs the most probable matches from the database.

6.3.2. Retrieving object models In order to find an object in the received image we have to generate a database document in the same way as described in the section above. We first extract the SIFT features from the received image and we quantize the descriptor vectors to words with the vocabulary tree. A single document is formed from all words of the input image and we can query the database with it. The database returns the best N matches with their respective scores s (between 0 and 2, where 0 is best and 2 is worst). A score is calculated by comparing the two documents (d_1, d_2) in a scoring function realized as an L2-norm metric:

$$s(d_1, d_2) = \left\| \frac{d_1}{\|d_1\|_2} - \frac{d_2}{\|d_2\|_2} \right\|_2 . \tag{3}$$

The above-mentioned $[0 \ldots 2]$ interval is obtained through metric scaling.

6.4. 3D model reconstruction

In this section, we lay out the 3D reconstruction methods we integrated in order to create complete object model hypotheses that can be used in grasping scenarios. The focus in this section does not lie on millimeter-accurate modeling of clusters, which is realistically not possible in a lot of cases due to sensor noise and large variability in the objects we encounter. Instead, we strive to improve the models generated from point cloud data with regards to two

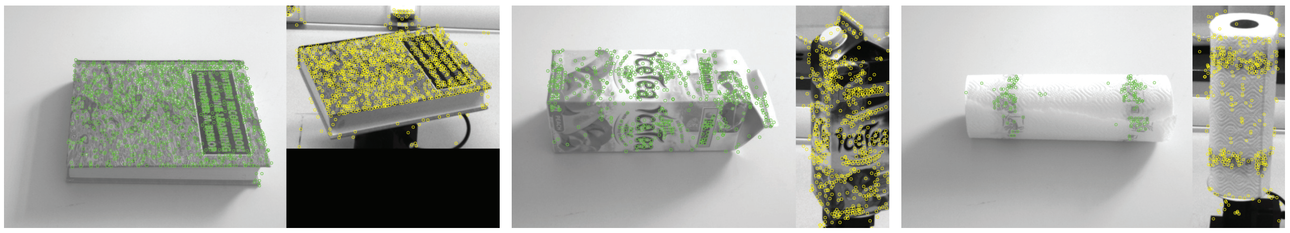


Fig. 13. Example of matching SIFT features for different objects from the imported database.

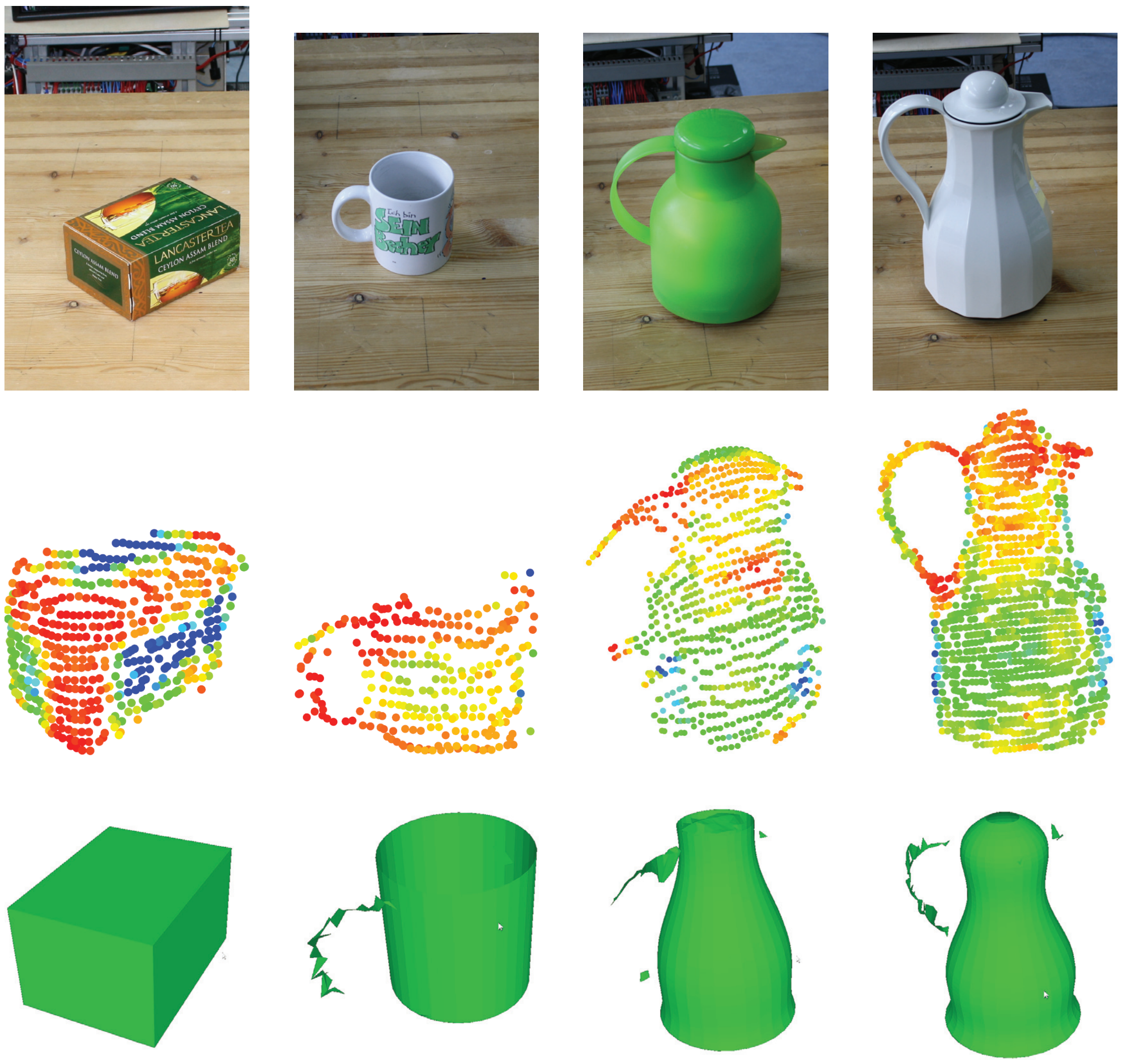


Fig. 14. Automatic reconstruction of objects of different types. The top row shows the estimated r_{min} values color-coded by the legend of Figure 9 (red 0, blue 0.2 m or higher).

aspects. First, we want the resulting reconstruction to be smooth, especially when compared with a simple meshing approach, where sensor noise causes problems in the grasp planner, e.g. when computing contact points and forces. On the other hand, we want to generate a working hypothesis concerning the backside of an object perceived only partially. Otherwise, grasp analysis might suggest a pinch grasp on the edge of the cluster, even though the object continues smoothly towards the unobserved back.

As of now, reconstruction algorithms have been developed for boxes, cylinders and rotational objects (Marton et al. 2010a), as shown in Figure 14. We chose these models because they are inherently symmetrical and occur frequently in a large number of objects (as shown in Table 1).

In the case of cylinders and rotational objects, a rotation axis is determined, which allows for simple completion of the object model, and for boxes, the hidden surfaces can also be retrieved relatively straightforward.

If a fitted model does not have at least 75% of the points as inliers, we assume that it has a more complex shape, and a simple triangulation is performed as a fall back. Also in the other cases, the outliers of the geometric models are triangulated and added to the final model as a mesh, e.g. to model additional geometric features of an object, such as handles. Figure 15 shows such an example. As the majority of grasp planning methods use triangular meshes, the geometric shapes are decomposed in triangles as well, but we are exploring the use of the shapes directly for more optimized grasping.

6.4.1. Box and cylinder fitting The decision on selecting the appropriate fitting model is based on our previous work on footprint analysis (Marton et al. 2009a), but, in addition, we rely on the 3D normals to robustly detect a circular or a rectangular footprint, also during the 3D model fitting and validation step. This enables our method to make the best choice regarding what model it should choose.

If a sufficient number of points have a minimum radius greater than 0.1 m they are highly likely to lie on planar surfaces and we set out to find the best fitting box to the cluster. Unlike Marton et al. (2009a), we fit a rectangular model directly to the points having normals perpendicular to the 'up' axis (as we assume boxes to be statically stable, i.e. standing on one of their sides) and maximize the number of inliers in a RANSAC loop. This direct approach estimates the box orientation and outperforms the line detection and merging and/or box detection based on PCA. Since we assume a correct segmentation and noise is removed in pre-processing, the box dimensions are computed from the oriented bounding box, and a final inlier count is performed.

For cylinders, we use a RANSAC approach which is based on the observation that on a cylinder surface, all normals are orthogonal to the cylinder axis, and intersect it. We consider the two lines defined by two sample points and their corresponding normals as two skew lines, and the shortest connecting line segment as the axis. Determining the radius is then a matter of computing the distance of one of the sample points to the axis.

The quality of box and cylinder fittings for two objects is presented in Figure 16. Further tests on a total of 267 measurements of 5 objects show an average fitting error of 5.31 mm with a standard deviation (SD) of 2.69 mm for the extensions of boxes, and slightly less for cylinder radii, an average of 2.85 mm and SD of 1.23 mm. Both are well below the original sensor noise of roughly 1 cm. The average error for the axis orientation for cylinders was 1.11° with a SD of 0.34°. By enforcing an upright position for the axes results are more robust, but it is not mandatory.

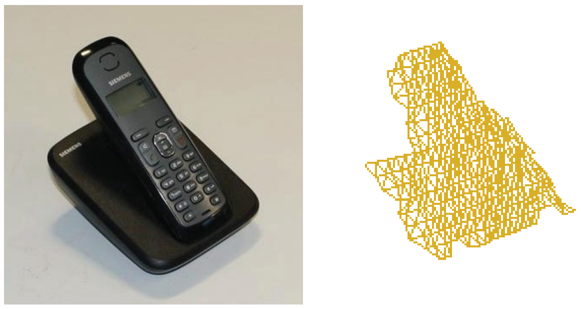


Fig. 15. Triangulation of an object where no good model could be fitted.

Compared with the original approach presented by Marton et al. (2009a) we can see a clear increase in accuracy. On the same scans, the obtained average errors for box extents was 10.6 mm with SD 6.05 mm. Results did not improve (nor worsen) for cylinders as the methods have the same basis, but instead we are now reconstructing them in arbitrary poses, while the original approach assumed all objects to be upright (and would consider lying cylinders to be boxes).

6.4.2. Estimation of rotational surfaces To reconstruct surfaces of revolution, we employ a RANSAC-based two-step approach as described by Blodow et al. (2009). In the first step, a rotation axis is estimated from sample points by minimizing a function over the line-to-line distances between the axis and the lines defined by each sample point and its corresponding normal. This is based on the observation that for a rotational object, a line constructed from a point and corresponding normal intersects the symmetry axis, similar to the cylinder case. The contour line is then estimated in the second step (as described in the following).

Let $\langle a, \vec{a} \rangle$ denote the axis, defined by a point a and a direction vector $\vec{a}$ and let $\langle p_i, \vec{n}_i \rangle$ denote the line defined by the ith sample point and its corresponding normal vector. Then, we minimize the following function over a and $\vec{a}$:

$$\sum_{i=0}^{m} d_{l,l}(\, \langle a, \vec{a} \rangle, \langle p_i, \vec{n}_i \rangle)^2 , \qquad (4)$$

where $d_{l,l}$ stands for the line-to-line distance. This can be solved using a non-linear optimizer such as Levenberg–Marquardt.

Once an axis has been found, the original sample points are transformed into a 2D coordinate system such that the rotation axis coincides with the x-axis. Every point p_i is projected onto a point $p_{i,2D}$ whose x coordinate is defined as its position along the rotation axis, and whose y coordinate represents the point-to-line distance between p_i and $\langle a, \vec{a} \rangle$.

We then employ a polynomial fitting step based on least-squares minimization to fit a preliminary contour line to

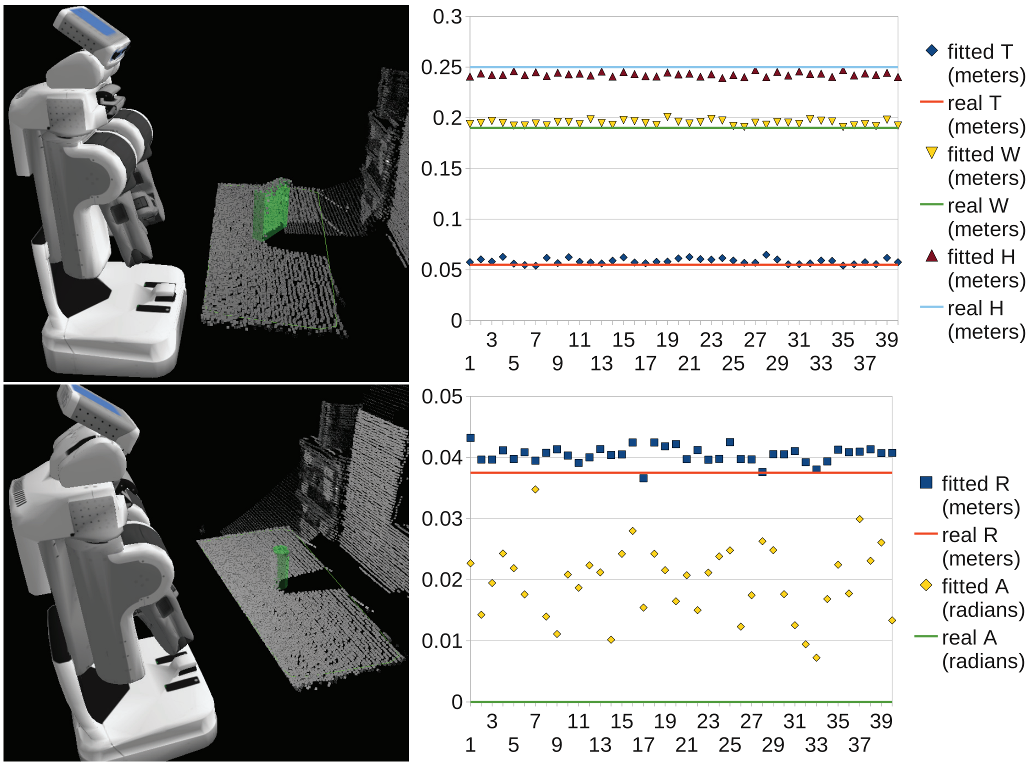


Fig. 16. Left: Visualization of the setup used for measuring fitting errors. The model and the table are marked with green. Right: Errors in estimated box width (W), height (H), thickness (T), and in estimated cylinder radius (R) and angle to the vertical (A). Graph shows these errors for subsequent scanning and fitting (x-axis shows the scan number).

the projected (2D) sample points. This contour line is then used to determine which points are inliers to the rotational model, and the polynomial is refitted using these inliers. This can be repeated until changing the polynomial coefficients does not increase the number of inliers. An example of this 2D problem is shown in Figure 17(right) for the teapot data set used earlier. Note that the contour curve is not influenced by the outliers that come from the handle, for example.

Unfortunately, the axis estimation step as presented in Blodow et al. (2009) is sensitive to higher degrees of noise, which we encounter with less accurate 3D sensors. The minimization function (4) becomes less smooth and the optimization becomes ‘stuck’ in local optima most of the time, as can be seen in Figure 17(left). We alleviated this problem by performing the rotational estimation on smooth resampled point cloud data, which has much less noise and more accurate normals (see Figure 17(middle)). Figure 18 shows the analysis of errors in determining the axis during rotation estimation.

This smoothing step alleviates some of the problems originating from noisy normal estimation, but it also loses detail on the underlying surface, such as sharp creases. The fitting step can therefore not be guaranteed to capture the ground-truth model. In our experiments, the standard deviation of the distances of points to the model were found to be of the order of 2 mm with a correlation coefficient of $r^2 \approx 0.95$, however this only represents how well the model fits the data points, not the underlying actual geometry. For a more detailed analysis, we have to refer to the previous work on these methods as the significance of the herein presented fitting methods lies in the *completion* of partially perceived models, and not in quality of fit of the reconstruction methods.

Although the rotational estimation is less accurate and computationally more intensive than fitting of simple models, it is an important addition in order to be able to deal with more types of objects (see Table 1). More accurate results can be obtained by increasing the expected probability of a successful fit for RANSAC (and thus the maximum number of iteration), but in order to obtain results in under 2 seconds, axis estimation errors around 10° were obtained.

In the case of box-like and cylindrical models, the fact that the dimensions are computed accurately from partial views already validates the use of this approach for grasp planning. As presented in Section 8.1 having a hypothesized backside makes a significant difference when the goal is to pre-compute realistic stable grasps. Similarly, this is an important consideration for rotational objects as well, as considering only the scanned points does not give any estimate on how large the object is, and how the backside looks. While we do not claim to have solved every

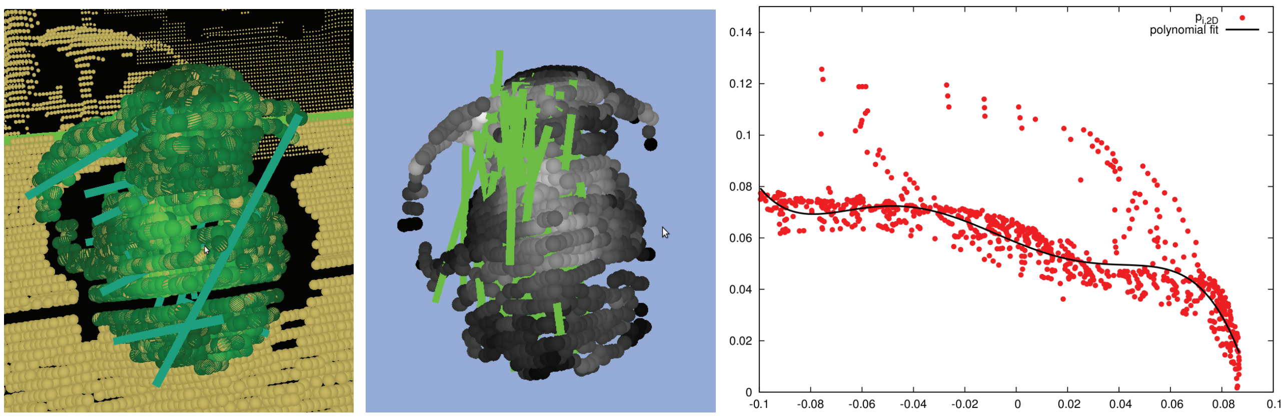


Fig. 17. The axis estimation step fails consecutively if surface points and normals contain too much noise (left). Re-sampling the points results in usable axis estimates (middle). Fitting a polynomial function to generate the contour curve given the axis (right).

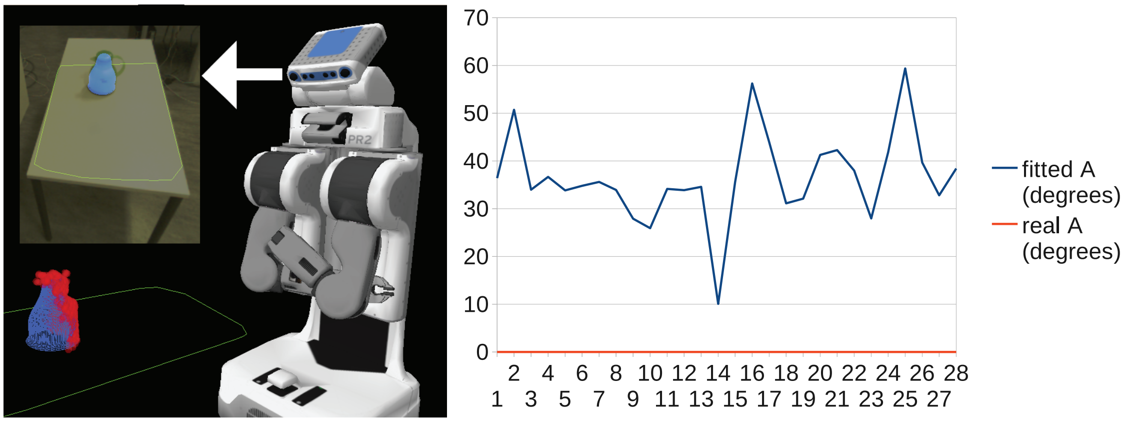


Fig. 18. Left: Visualization of the setup used for measuring fitting errors. Right: Errors in the angle of the estimated axis to the vertical (A) if no visibility checks are performed in the optimization loop. The inaccurate fits are penalized by their parts in free space, and the more accurate fits, such as at iteration 14, will be selected as the final model. Graph shows these errors for subsequent axis optimization steps, at different RANSAC iterations.

problem related to this uncertainty, approaches such as ours or that of Thrun and Wegbreit (2005) provide useful hypotheses. Their full evaluation, apart from the fit quality presented above, is on our agenda, including the acquisition of enough ground-truth data, performing and evaluating the grasping, and the analysis the effect of fitting errors on grasp stability when compared with reactive blob-grasping.

7. System evaluation

To facilitate object recognition we maintain a database of the descriptors provided by the examination methods. It is important to note that not all descriptors must be known due to the multimodal nature and incremental learning feature of our system. The view-dependent variations of the descriptors are stored for every view, thus each view of an object that has been observed can be looked up and compared with object hypotheses during classification. The examination methods described in Section 6 provide the following data on a per-view basis: the geometric category, the appearance descriptors, the model parameters and the 3D mesh. The timestamped position of the last detection is saved for each object as well.

The following sections provide details of the object identification and model learning modules using the above-presented subsystems. We have to differentiate between the two operating modes, *active* perception, when a given object needs to be located, and *passive* perception of objects for updating their locations and models. Since the *object model database* does not have to contain every possible piece of information about the objects, it is straightforward to look for a novel object by adding a new entry that contains known information about the object.

7.1. *Object classification*

For object classification, we iterate over the examination methods (Figure 5), run them on the object hypotheses to extract features (if possible), and search through the list of available objects in the database to see which ones present the same or similar features. The objects that match are kept in the list for the next iteration of comparison, and the others are removed. In this way, at each step the set of possible objects is reduced. As discussed previously, if multiple objects remain after checking all features, the result is deemed ambiguous, whereas if at a given step no objects match, we consider the object hypothesis to be a novel object.

Owing to the combination of features, an object candidate is accepted only if it matches all of the checked features. This reduces the effect of misclassifications, at the cost of observed views of known objects presenting previously unobserved features being added as new objects (unless observed in subsequent scans). As the preceding process is started only if the object was not identified in the previous step (otherwise the previous identity is assumed), new views can be learned through continuous observation of an object as presented in the next section.

Figure 19 shows results we obtained with our first batch of 12 testing objects. For the 3D categorization we used the model trained as described in Section 6.2 and the visual features were trained on the SemanticDB database as detailed in Section 6.3. The tests show correct classifications, achieved in under 2 seconds (not considering acquisition time). Please see the following sections for a more detailed evaluation.

7.2. Detection based on known views

In order to evaluate our approach as a whole, we performed the detection and recognition test in our kitchen laboratory (see the left column of Figure 20). The test was carried out on a total number of 13 objects located at 4 different scenes (denoted with Scene 1 to Scene 4 and depicted in top-down order in the right column of Figure 20). The robot was programmed to navigate to each of the scenes and capture point cloud and image from several different views by traversing along the free paths around the scenes. The basic *planar support* approach could not have been applied for the Scenes 2 and 4 as the supporting planes are too high, thus impossible to scan with either of our robots.

The vocabulary tree and corresponding database with descriptors were trained and built from images from the SemanticDB database and 10 more images of products from the GermanDeli[13] Web site. The parameters K and L were both set to 5, resulting in a 1 minute training time of the database for the 65,000 features extracted from 170 images. In this configuration the querying for one object cluster took 50 ms. Setting the score value of the database retrieval mechanism to the experimentally determined value of 1.0 enables us to classify all measurements that exceed this value as unknown.

The SIFT-based detection performed well (Table 2), and geometric classification achieved just below 80% success rate (lower than in the experiments described earlier due to partial occlusions). The final database that was built contained some mistakes, as mentioned in Section 1, which required a few manual corrections, but all in all constituted less effort than manually adding all descriptors of all new object to the database using a rotating table for example.

7.3. Improved detection through incremental learning

In the case that an object hypothesis is detected in the same position in the map's coordinate frame in subsequent scans we assume it is the same object as that identified previously. For the localization we use an AMCL-based framework combining robot's odometry and laser readings in *a priori* built 2D map (Pfaff et al. 2006). Localization's absolute error margin lies at 0.02 m on average, thus we consider object hypotheses to represent the same object if they are not further away than 0.05 m. If the subsequent call of the examination methods returns no matching views for the given object hypothesis, we store the current observation as a new view of the object in the *object model database*, as shown in Figure 21.

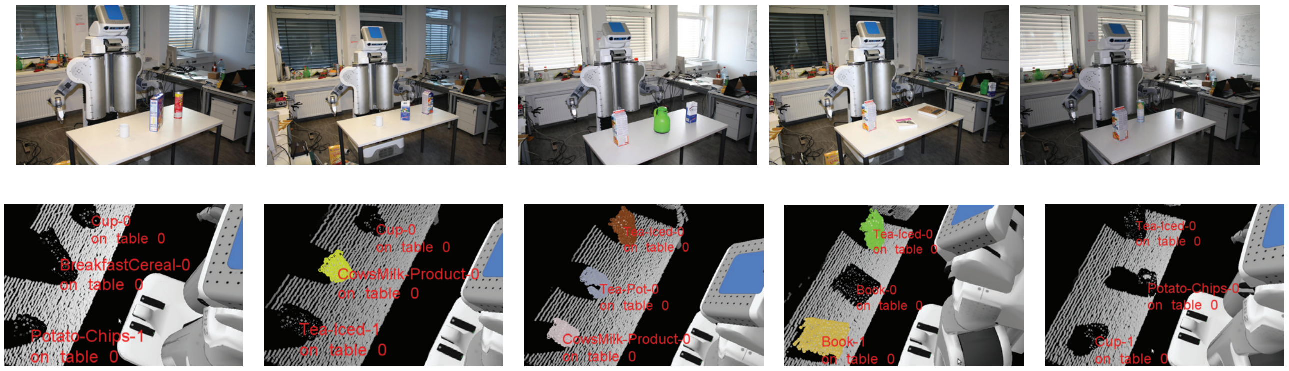
To demonstrate the capability of our system to acquire new object models on the fly we set up the Scene 1 with one unknown object (green milk box) which in fact generated all 10 unclassified views reported in the first row of Table 2. Knowing that these do not match anything in the database, we can introduce them as new object models. The assumption we are making here is that the scene remains static, thus the cluster cloud and the defined ROI at the given 3D position in the world coordinate frame contain the same object.

After this we performed another test run on the Scene 1 with the re-trained tree and the updated database of SIFT descriptors and were able to reduce the number of not detected objects down to two as shown in Table 3.

Since most large databases (e.g. the GermanDeli Web site) offer only single pictures of objects, incremental learning is an important feature for a perception system that needs to develop over time.

7.4. Discussion on failure cases

All five SIFT-based classification failures for Scene 2 (second row of Table 2 and right column of Figure 20) are attributed to the reflective metallic surface of a green chips cylinder which made the classification using SIFT descriptors practically impossible in that position. And this is where the power of our multimodal system becomes apparent. In all six measurements of this object for this scene it was correctly categorized as a *tube-like cylinder*, which for the set of our 13 objects made the *perceptual task execution*



(a) cup-0
Break fast cereal-0
Potato Chips-1

(b) Cup-0
CowsMilk-
Product-0
Tea-Iced-1

(c) Tea-Iced-0
Tea-Pot-0
CowsMilk-
Product-0

(d) Tea-Iced-0
Book-0
Book-1

(e) Tea-Iced-0
Potato-Chips-0
Cup-1

Fig. 19. Classification results for a variety of objects. Clusters shown in random colors.

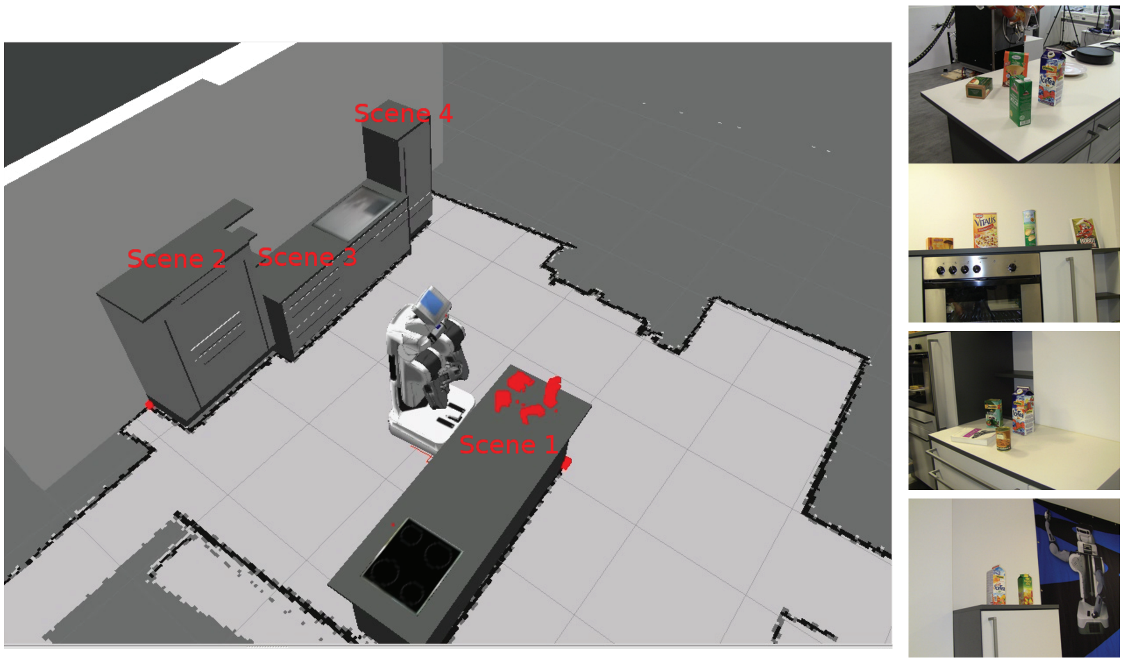


Fig. 20. We performed the final evaluation test on a total number of 13 objects located at 4 different scenes in our kitchen lab (denoted with Scene 1 to Scene 4 and depicted in top-down order in the right column). The robot was programmed to navigate to each of the scenes and capture point cloud and image from several different views by traversing along the free paths around the scenes.

Table 2. Detection of objects and identification of unknown views using SIFT with vocabulary trees.

Scene	#Views/#Known	#Failures	#Unknown	Success (with/no unknowns)
Scene 1	52/42	0	10	80.8%/100%
Scene 2	11/11	5	0	54.5%/54.5%
Scene 3	24/24	2	0	91.6%/91.6%
Scene 4	12/12	0	0	100%/100%
Total	99/89	7	10	82.8%/92.1%

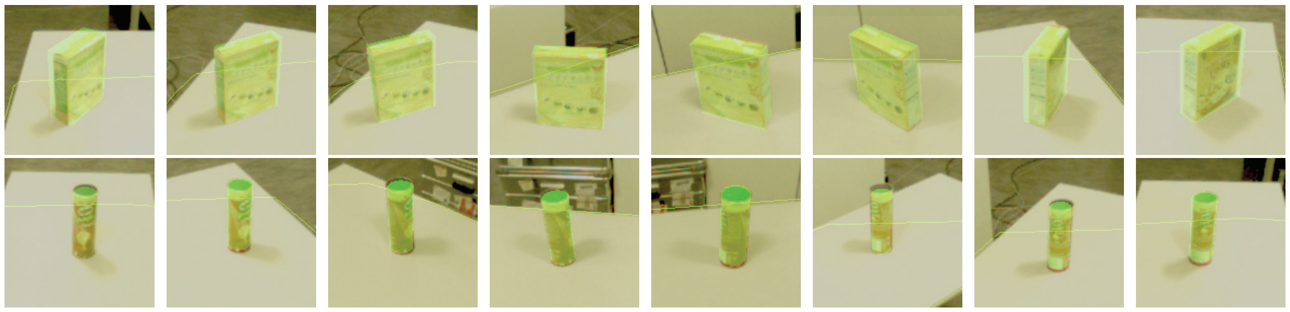


Fig. 21. Different views of a cereal box (top) and chips (bottom) learned by multiple observations in the same position. The green line represents the table's edge as detected by the laser (some parts of it were not scanned), and the green markers denote the projections of the fitted 3D model onto the image data.

select the green chips cylinder object as the only tall cylindrical object in the *object model database*. In a database with a larger number of objects containing, for example, multiple tall cylinders correct geometric categorization would still provide a significant hint at the possible class instance. In addition, the geometric model reconstruction provides the means for eventual manipulation.

8. Application scenarios

Object reconstruction, categorization and classification routines are useful for addressing many practical problems in robotics. Despite these not being the main focus of the paper we would like to briefly present the use cases for the presented system here.

8.1. Grasping of objects using reconstructed object models

For grasping applications the geometric models are the most important ones, thus if an object needs to be grasped the geometric reconstruction has to be performed as denoted in Section 6.4. The triangulated meshes are then compared with stored models which have already grasp points computed, and if no good enough match was found they are uploaded into the GraspIt simulator (Goldfeder et al. 2009) where they get annotated with large sets of stable, pre-computed grasp points. Computed grasps can then be performed by the robot, as shown in Figure 22

Grasps for novel objects (e.g. Figure 15) are computed for the specific robot (PR2 and TUM-Rosie) at runtime from 3D sensor data, using heuristics based on both the overall shape of the object and its local features. We kindly refer the reader to the papers by Hsiao et al. (2010) and Marton et al. (2010a) to obtain more information for PR2 and TUM-Rosie calculation of grasps.

The major benefit of the reconstruction routine is that grasps are calculated by also considering the backsides of the objects as depicted in Figure 23. From the contacts, the grasp quality is statically estimated using the algorithm presented by Ferrari and Canny (1992).

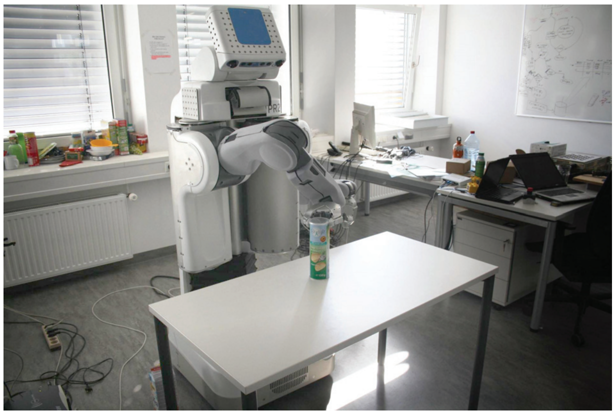


Fig. 22. Grasping of the localized object by the PR2 robot. The fitted geometric model was triangulated and fed into the grasp planning software to obtain grasping points.

8.2. CAD model-based object detection

It is possible to perform 6-DOF position retrieval of objects on camera images using state-of-the-art 3D shape model matching technique that simulates the 2D appearance of the objects in a shape model generation phase (Ulrich et al. 2009). The routine receives object candidates (clusters of points) projected onto the corresponding 2D image from point cloud data and then uses *a priori* built CAD model of objects to perform the actual detection. The strength of this approach is based on the fact that the CAD models do not need to be modeled manually but can rather be generated by the robot itself in the desired complexity (see Figure 24 for the matching results and Section 6.4 for the generation of CAD models).

These meshes can be used by other systems as well, even if the sensors and their setup is different, as detailed in the following section. If, however, the model is built by the agent performing the CAD model-based detection, the known pose of the model can be used to avoid unnecessary matching of very different poses in the image.

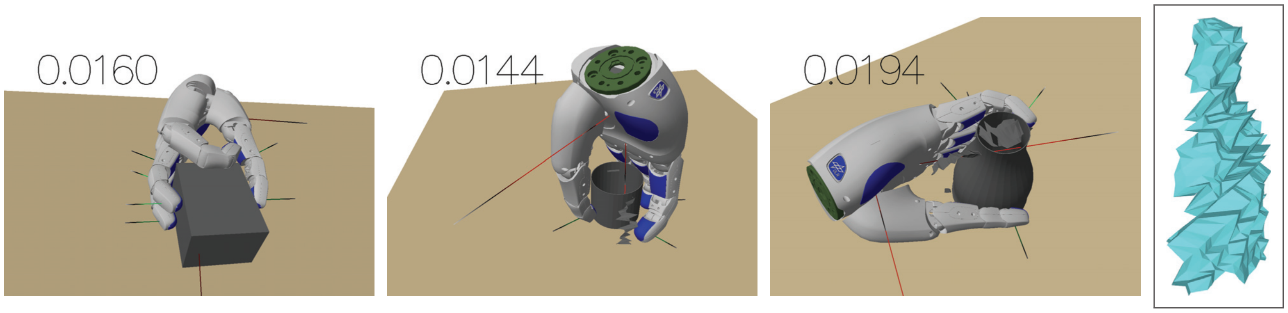


Fig. 23. Grasp planning application for TUM-Rosie. Note that we used incomplete scans of the objects, such as that depicted in the last image (side view of the triangulated mesh) and were still able to generate good grasps for the backsides.

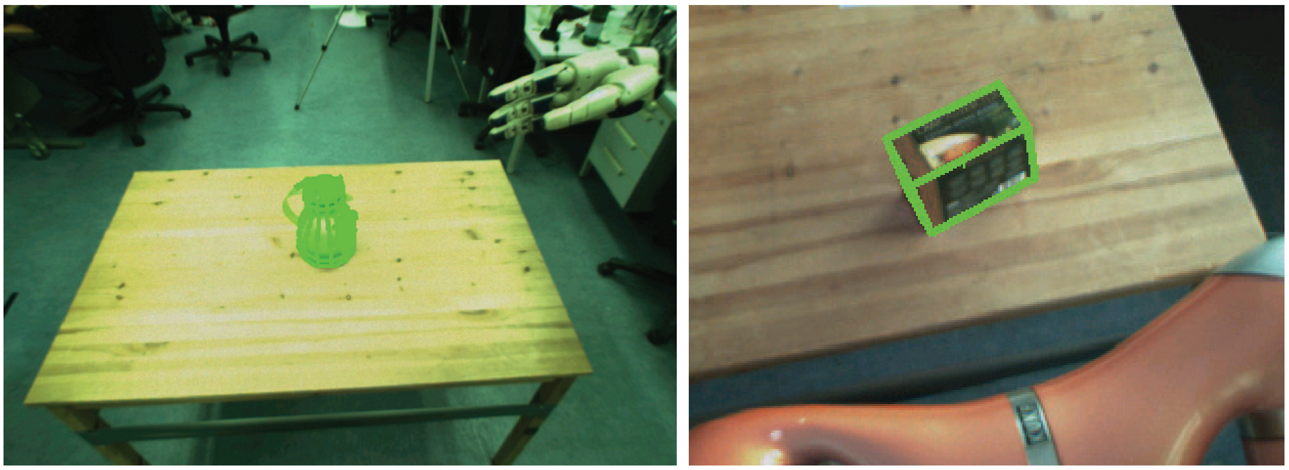


Fig. 24. Locating the resultant geometric model in an image by the TUM-Rosie robot.

Table 3. Improved detection for Scene 1 from Figure 20 before and after the vocabulary tree was re-trained and database rebuilt with the templates for green milk box. All in all, more views received the correct label.

Scene 1	#Views/#Known	#Failures	#Unknown	Success (with/no unknowns)
Before	52/42	0	10	**80.8%**/100%
After	52/52	2	0	**96.2%**/96.2%

8.3. Lifelong dynamic world state mapping

The multimodal nature of our perception system lends itself to applications for entity resolution (Blodow et al. 2010), where we proposed to use Markov logic networks to probabilistically track trajectories of objects over time using reconstructions and refinement algorithms as described in Section 6. In principle, the approach computes similarity measures between different object cluster observations from compatible object descriptions (e.g. category and/or SIFT descriptors) and uses the full power of first-order logic with the probabilistic semantics of graphical models in order to deal with difficult conditions such as partial observability of the robot's environment, incomplete object descriptions, and a dynamic environment in which we cannot observe the human and his actions directly.

8.4. Platform independent perception system

Our perception system has been validated on two state-of-the-art service robots, the TUM-Rosie and the PR2 thanks to the general infrastructure provided by ROS. Since the only prerequisite is to have the kinematic chains between the 2D cameras and the 3D tilting laser sensor calibrated with the precision under 5 mm (which still ensures correct ROI extraction for the purpose of tabletop manipulation), our system can thus be ported to any arbitrary robot platform that meets the above requirement. As most of the testing for this article was performed with the PR2 robot, Figures 23 and 24 verify our claims for the TUM-Rosie robot by showing the execution of the shape-based matching algorithm and grasp planning, respectively.

While our two robots have similar sensor setups, cameras and lasers are widely used on different platforms as well, and many 3D sensors provide accuracy similar to or better than the Hokuyo UTM-30LX, ensuring the usability of the presented methods.

9. Conclusions and future work

In this paper, we have presented a system for automatic and efficient object detection and modeling that can improve its models with limited human supervision. The high number and variety of objects in a typical household require the use of multiple descriptive features and the automatic construction of geometric models for performing robotics tasks autonomously, as we believe that having an object database built by the robot itself in the operating environment (or adapting/enriching a seed database) is more suitable to deal with the specifics of the environment efficiently than a generic approach of providing as many models of as many objects as possible.

In the future, we plan to automatically learn the objects' geometric classes and learn which geometric model type fits best to each class. Also, after multiple views of an object are obtained, the geometric models should be re-adapted into a consistent global model which should be used to locate the object in the scenes. In this way, the handle of a cup could be assumed to be in the back even without being detected.

Probably the most important and exciting application of the presented methods would be the extension of the system with more features and the integration with the system presented in Section 8.3. This advancement would bring household robot assistants one step closer to being truly autonomous when it comes to dealing with objects of daily use. The integration of multiple sources of information, prior knowledge, and combining multiple observations seem to be natural ways to achieve robust performance.

Notes

1. See http://i61p109.ira.uka.de/ObjectModelsWebUI/.
2. See http://www.ros.org.
3. See http://www.pointclouds.org.
4. See http://docs.pointclouds.org/trunk/classpcl_1_1_r_s_d_estimation.html.
5. See http://code.cs.tum.edu/indefero/index.php//p/mapping/source/tree/master/pcl_cloud_algos/src/pcl_cloud_algos/global_rsd.cpp.
6. See http://www.ros.org/wiki/objects_of_daily_use_finder.
7. See http://www.ros.org/wiki/urdf.
8. Since we do not assume that normals are oriented consistently we compute α as the angle between the lines defined by the two points and their normals.
9. See http://ias.in.tum.de/kb/wiki/index.php/Example_images_of_kitchens.
10. See http://ias.cs.tum.edu/download/semantic-3d.
11. See http://www.ros.org/wiki/household_objects_database.
12. See http://sourceforge.net/projects/libsift/.
13. See http://www.germandeli.com.

Funding

This work was supported by the DFG cluster of excellence CoTeSys (Cognition for Technical Systems) at the Technische Universität München, and by Willow Garage, Menlo Park, CA. Zoltan Csaba Marton gratefully acknowledges the support of the TUM Graduate School's Thematic Graduate Center Cognition for Technical Systems at the Technische Universität München.

Acknowledgements

The authors would like to thank Ulrich Klank and David Weikersdorfer for their help.

References

Arbeiter G, Fischer J, and Verl A (2010) 3D Perception and Modeling for Manipulation on Care-O-bot® 3. In *Proceedings of IEEE/Robotics and Automation Society: 2010 IEEE International Conference on Robotics and Automation (ICRA 2010): Conference Digest*, Anchorage, Alaska, USA, May 3–8, 2010. Piscataway, NJ, USA: IEEE Press, 2010, 5S.

Biegelbauer G and Vincze M (2007) Efficient 3D object detection by fitting superquadrics to range image data for robot's object manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 2007.

Blodow N, Jain D, Marton Z-C and Beetz M (2010) Perception and probabilistic anchoring for dynamic world state logging. In *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, 6–8 December 2010.

Blodow N, Rusu RB, Marton ZC and Beetz M (2009) Partial view modeling and validation in 3D laser scans for grasping. In *9th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Paris, France, 7–10 December 2009.

Bone GM, Lambert A and Edwards M (2008) Automated modeling and robotic grasping of unknown three-dimensional objects. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, 2008.

Chang C-C and Lin C-J (2001) *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Choi YS, Deyle T and Kemp CC (2009) A list of household objects for robotic retrieval prioritized by people with ALS (Version 092008). *CoRR*, DOI: abs/0902.2186.

Coates A, Baumstarck P, Le QV and Ng AY (2009) Scalable learning for object detection with GPU hardware. In *Proceedings of IROS 2009*, pp. 4287–4293.

Fergus R, Perona P and Zisserman A (2003) Object class recognition by unsupervised scale-invariant learning. In *Proceedings of CVPR*, Wisconsin, USA, 2003, pp. 264–271.

Ferrari C and Canny J (1992) Planning optimal grasps. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, May 1992.

Fritz M, Darrell T, Black M, Bradski G and Karayev S (2009) An Additive Latent Feature Model for Transparent Object Recognition. Oral presentation at *NIPS*, December 2009.

Goldfeder C, Ciocarlie M, Dang H and Allen PK (2009) The Columbia grasp database. In *IEEE International Conference on Robotics and Automation, 2009 (ICRA '09)*, Kobe, Japan, May 2009, pp. 1710–1716.

Hillenbrand U (2008) Pose clustering from stereo data. In *Proceedings VISAPP International Workshop on Robotic Perception (RoboPerc 2008)*.

Hinterstoisser S, Lepetit V, Ilic S, Fua P and Navab N (2010) Dominant orientation templates for real-time detection of texture-less objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

Horswill I (1995) Integrating vision and natural language without central models. In *Proceedings of the AAAI Fall Symposium on Embodied Language and Action*, 1995.

Hsiao K, Chitta S, Ciocarlie M and Jones EG (2010) Contact-reactive grasping of objects with partial shape information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

Johnson AE (1997) *Spin-images: A Representation for 3-D Surface Matching*. PhD thesis. Technical Report, 1997, CMU-RI-TR-97-47, Robotics Institute, Carnegie Mellon University.

Klank U, Pangercic D, Rusu RB and Beetz M (2009) Real-time cad model matching for mobile manipulation and grasping. In *9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France, 7–10 December 2009, pp. 290–296.

Kragic D and Vincze M (2009) Vision for robotics. *Found Trends Robot* 1: 1–78.

Lai K and Fox D (2009) 3D laser scan classification using web data and domain adaptation. In *Proceedings of Robotics: Science and Systems*, Seattle, WA, June 2009.

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *Int J Comput Vis* 60: 91–110.

Marton ZC, Goron L, Rusu RB and Beetz M (2009a) Reconstruction and verification of 3D object models for grasping. In *Proceedings of the 14th International Symposium on Robotics Research (ISRR09)*, Lucerne, Switzerland, 31 August–3 September 2009.

Marton Z-C, Pangercic D, Blodow N, Kleinehellefort J and Beetz M (2010a) General 3D Modelling of Novel Objects from a Single View. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 18-22 2010a.

Marton Z-C, Pangercic D, Rusu RB, Holzbach A and Beetz M (2010b) Hierarchical object geometric categorization and appearance classification for mobile manipulation. In *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, 6–8 December 2010.

Marton ZC, Rusu RB and Beetz M (2009b) On fast surface reconstruction methods for large and noisy datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 12–17 May 2009.

Miller A and Allen PK (2004) Graspit!: A versatile simulator for robotic grasping. *IEEE Robotics Automat Mag* 11: 110–122.

Nakayama H, Harada T and Kuniyoshi Y (2009) AI Goggles: Real-time description and retrieval in the real world with online learning. In *Canadian Conference on Computer and Robot Vision*, pp. 184–191.

Nister D and Stewenius H (2006) Scalable recognition with a vocabulary tree. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC: IEEE Computer Society, pp. 2161–2168.

Pangercic D, Tenorth M, Jain D and Beetz M (2010) Combining perception and knowledge processing for everyday manipulation. In *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, Taipei, Taiwan, 18–22 October 2010.

Pfaff P, Burgard W and Fox D (2006) Robust Monte-Carlo localization using adaptive likelihood models. In *Proceedings of EUROS*, pp. 181–194.

Quigley M, Batra S, Gould S, Klingbeil E, Le QV, Wellman A and Ng AY (2009) High-accuracy 3D sensing for mobile manipulation: Improving object detection and door opening. In *Proceedings of ICRA*, Kobe, Japan, 2009.

Richtsfeld M and Vincze M (2008) Grasping of unknown objects from a table top. In *Workshop on Vision in Action: Efficient Strategies for Cognitive Agents in Complex Environments*, 2008.

Robertson S (2004) Understanding inverse document frequency: On theoretical arguments for IDF. *J Documentation* 60(5), 503–520.

Romea AC, Berenson D, Srinivasa S and Ferguson D (2009) Object recognition and full pose registration from a single image for robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA '09)*, May 2009.

Ruhnke M, Steder B, Grisetti G and Burgard W (2009) Unsupervised learning of 3d object models from partial views. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

Rusu RB, Blodow N, Marton ZC and Beetz M (2009a) Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in human environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, 11–15 October 2009.

Rusu RB, Holzbach A, Bradski G and Beetz M (2009b) Detecting and segmenting objects for mobile manipulation. In *Proceedings of IEEE Workshop on Search in 3D and Video (S3DV), held in conjunction with the 12th IEEE International Conference on Computer Vision (ICCV)*, Kyoto, Japan, 27 September 2009.

Rusu RB, Marton ZC, Blodow N and Beetz M (2008a) Learning informative point classes for the acquisition of object model maps. In *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Hanoi, Vietnam, 17–20 December 2008.

Rusu RB, Marton ZC, Blodow N, Dolha M and Beetz M (2008b) Towards 3D point cloud based object maps for household environments. *Robotics Autonomous Syst J*, 56(11): 927–941.

Saxena A, Driemeyer J and Ng AY (2008) Robotic grasping of novel objects using vision. *Int J Robotics Res* 27: 157–173.

Schnabel R, Wahl R and Klein R (2007) Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum* 26: 214–226.

Sivic J, Russell BC, Efros AA, Zisserman A and Freeman WT (2005) Discovering object categories in image collections. In *Proceedings of the International Conference on Computer Vision*, 2005.

Steder B, Grisetti G, Van Loock M and Burgard W (2009) Robust on-line model-based object detection from range images. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, October 2009.

Strobl KH, Mair E, Bodenmüller T, Kielhöfer S, Sepp W, Suppa M, et al. (2009) The self-referenced DLR 3D-modeler. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, October 2009, pp. 21–28.

Thrun S and Wegbreit B (2005) Shape from symmetry. In *Proceedings of the International Conference on Computer*

Vision (ICCV), Bejing, China, 2005. Piscataway, NJ: IEEE Press.

Torres MM, Romea AC and Srinivasa S (2010) MOPED: A scalable and low latency object recognition and pose estimation system. In *Proceedings of ICRA 2010*, May 2010.

Triebel R, Shin J and Siegwart R (2010) Segmentation and unsupervised part-based discovery of repetitive objects. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

Ulrich M, Wiedemann C and Steger C (2009) CAD-based recognition of 3D objects in monocular images. In *International Conference on Robotics and Automation*, pp. 1191–1198.

Wurm KM, Hornung A, Bennewitz M, Stachniss C and Burgard W (2010) OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, May 2010. Software available at http://octomap.sf.net/.

Wyrobek K, Berger E, Van der Loos HFM and Salisbury K (2008) Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In *of the International Conference on Robotics and Automation (ICRA)*, 2008.

Xue Z, Kasper A, Zoellner MJ and Dillmann R (2009) An automatic grasp planning system for service robots. In *International Conference on Advanced Robotics, 2009 (ICAR 2009)*, Munich, Germany, June 2009, pp. 1–6.

Zhang Y, Koschan A and Abidi MA (2004) Superquadric representation of automotive parts applying part decomposition. *Journal of Electronic Imaging* 13: 411–417.