Model-Based Furniture Recognition for Building Semantic Object Maps

Martin Günther[a,*], Thomas Wiemann[a], Sven Albrecht[a], Joachim Hertzberg[a,b]

[a] *Institute of Computer Science, Osnabrück University, Albrechtstr. 28, 49076 Osnabrück, Germany*
[b] *DFKI Robotics Innovation Center, Osnabrück Branch, 49076 Osnabrück, Germany*

Abstract

This paper presents an approach to creating a semantic map of an indoor environment incrementally and in closed loop, based on a series of 3D point clouds captured by a mobile robot using an RGB-D camera. Based on a semantic model about furniture objects (represented in an OWL-DL ontology with rules attached), we generate hypotheses for locations and 6DoF poses of object instances and verify them by matching a geometric model of the object (given as a CAD model) into the point cloud. The result, in addition to the registered point cloud, is a consistent mesh representation of the environment, further enriched by object models corresponding to the detected pieces of furniture. We demonstrate the robustness of our approach against occlusion and aperture limitations of the RGB-D frames, and against differences between the CAD models and the real objects. We evaluate the complete system on two challenging datasets featuring partial visibility and totaling over 800 frames. The results show complementary strengths and weaknesses of processing each frame directly vs. processing the fully registered scene, which accord with intuitive expectations.

Keywords: Semantic map, incremental mapping, closed-loop mapping, model-based object recognition, 3D point cloud, CAD model matching,

---

*corresponding author

Email addresses: `martin.guenther@uni-osnabrueck.de` (Martin Günther), `thomas.wiemann@uni-osnabrueck.de` (Thomas Wiemann), `sven.albrecht@uni-osnabrueck.de` (Sven Albrecht), `joachim.hertzberg@uni-osnabrueck.de` (Joachim Hertzberg)

OWL-DL ontology

1. Introduction

1.1. Closed-loop Incremental Semantic Mapping

Building 3D maps of indoor environments by mobile robots has received increasing interest since the launch of inexpensive 3D sensors such as the Microsoft Kinect. Several successful approaches exist that generate 3D point cloud maps (e.g., [1, 2]) or mesh representations (e.g., [3]) based on RGB-D data. Yet, automatically providing additional *semantic* information to the maps, such as location and type of furniture present, is still not well understood. Information on a semantic level, however, is necessary for many advanced tasks of an autonomous robot, such as object search or place recognition. Also, it has advantages for the map building process itself: If the class and location of objects in the map are known, object models could be used to hypothesize missing sensor data, or loop-closing in mapping can be based on semantic as well as geometric information.

A semantic map is necessarily hybrid in the classical sense of Kuipers [4], including at least geometric information and semantic knowledge [5]. Intuitively, the process of generating a semantic map (semantic mapping, for short) should be closed-loop and incremental. *Closed-loop* means that object recognition and labeling in the sensor data ("bottom-up") should not strictly precede processing on the semantic level, but that knowledge and reasoning on the semantic level should be able to influence object classification, recognition and the mapping process as a whole ("top-down"). For example, the information that some room is an office room should lead reasoning on the semantic level to hypothesize that certain types of objects are likely or unlikely to be present, respectively; such hypotheses then bias or guide bottom-up sensor data processing. *Incremental* means that the semantic map building process does not have to wait for the sensor data of some scene or environment to be complete (no matter how such completeness would have to be defined and determined), but has to start right away, based on individual sensor takes, such as single RGB-D frames or 3D laser scans. This expectation fits with the closed-loop property, as the increase of environment knowledge in the semantic map over the mapping process is supported by both sensor data and prior knowledge, e.g., about object classes and their relations. Incrementality poses a challenge, though, as such individual sensor takes suffer greatly from occlusions and limitations due to sensor aperture or view pose constraints – in addition to the unavoidable regular sensor noise.

Closed-loop, incremental semantic mapping is currently not well understood. There is quite some body of literature about its ingredients, as will be discussed in the Related Work section; however, there are only few systems doing it in integration. This paper contributes a detailed case study of such a system. It presents an approach to semantic mapping that:

1. reconstructs the surfaces from noisy 3D data, captured from a Kinect camera, and creates a triangle mesh;
2. recognizes furniture objects in the point clouds based on structural descriptions from an OWL-DL ontology;
3. and finally adjusts their poses using ICP, and augments the created map with CAD models corresponding to the furniture objects.

We call *model-based object recognition* the ensemble of these three steps, used in integration with a knowledge base (given in OWL-DL, in this case) and a module for building geometric 3D point cloud maps. Using state of the art SLAM algorithms, these annotated point clouds can then be used to maintain a consistent semantic map of the complete environment, consisting of both the geometry and the semantic knowledge.

We would like to emphasize the role that using a formally well-understood knowledge representation and reasoning (KR&R) formalism plays in closed-loop, incremental semantic mapping, rather than using some ad-hoc set of object labels. Using arbitrary labels like "table", "tasse", or "q17", which may or may not have a meaning for humans, may suffice for purely bottom-up recognition, classification, or labeling of segments of sensor data. Whenever the intention is to reason with and about objects or events perceived by the robot, though, using some KR&R formalism with a well-defined semantics and, ideally, efficient reasoners available is the obvious choice. Such reasoning is needed for the top-down part of closed-loop semantic mapping in the first place; it may be employed in other robot tasks using the previously acquired semantic map, such as object search (e.g., [6, 7]), human robot interaction on a high conceptual level [8], detecting norm violations [9].

Many researchers in semantic mapping have recently been using description logics (DL, [10]) as such a formalism, in particular the DL variants OWL-DL and OWL-lite, as available in the OWL W3C standard [11]; we are using OWL-DL in the work reported here, too. DL is an obvious choice for a KR&R formalism in semantic mapping, as it allows to represent and reason about object ontologies, providing a structured representation of object classes and instances, but of some relations between objects, too, which the declarative part of a semantic map is expected to contain. Existing DL reasoners provide reasoning services such as consistency checking and subsumption within an ontology for free and in a highly optimized way. They allow sound inferences to be made across all hierarchical levels of the ontology without further effort. For example, questions like "How many pieces of furniture does room R contain?", or "Which pieces of furniture on this floor are suitable storage places for a milk jug?" could be answered right away, based on perceptions of individual chairs, tables, shelves and so on. As many representation and reasoning problems in robotics naturally include uncertainty, such as by sensor noise and/or interpretation uncertainties, several researchers have recently embedded the ontological reasoning provided by DL reasoners into probabilistic frameworks like Markov Logic Networks [12, 13] or Bayesian Logic Networks [14]. The bottom line here is:

1. Using a well-founded KR&R formalism for representing and reasoning in the semantic part of a semantic map is strongly advised, if not needed, in semantic mapping; previous AI work in KR&R has yielded a wide variety of such formalisms that are ready to be used.
2. Variants of DL have been used in much of the recent semantic mapping research, and we have done so in the work reported here.

"Pure" DL is certainly not the final word regarding a suitable KR&R formalism, as it cannot well handle uncertainty and n-ary relations, just to mention two points. Identifying or developing more fitting formalisms is an important issue for interdisciplinary research between AI and Robotics, which we recommend to put on the common agenda, but do not intend to detail in this paper.

1.2. Contribution of this Paper

The approach detailed in this paper combines a number of techniques that make it suitable for being used in on-line, incremental semantic mapping, starting from a stream of RGB-D frames: meshing of 3D points and referring to geometric features are used to compensate sensor noise and aperture limitations, and early closed-loop usage of the semantic knowledge is used to generate object hypotheses for guiding low-level sensor data processing.

The techniques are equally applicable to registered sets of RGB-D frames covering large areas and, hence, larger objects not captured in single frames.

In previous publications, we have presented partial results of this case study using 3D laser scanner data [15] and single RGB-D frames [16]. This paper presents an extended version of these conference publications. Additionally, we include new comprehensive results about what effect the degree of similarity between CAD model and actual object has on the quality of the final pose estimation, and we apply the method to a full scene point cloud using a 6D SLAM algorithm. The system implementing the method is evaluated on two different datasets consisting of a total of 810 point clouds containing six different classes of furniture. The complete data sets are available at `http://kos.informatik.uni-osnabrueck.de/furniture_recognition/`.

The paper next discusses related work, which is substantial in particular with respect to the individual parts of robotic mapping and object recognition. We then describe model-based object recognition in detail. After that, evaluation results are presented and discussed. To conclude, we sketch future work motivated by the findings of this case study.

2. Related Work

In the field of semantic mapping, several authors have proposed algorithms that label point clouds with semantic information. For instance, Rusu et al. [17] detect several types of furniture and approximate them as cuboids. Nüchter and Hertzberg [5] classify coarse structures (walls, floors, doors) using a semantic network and detect smaller objects using a trained classifier. Mason and Marthi [18] autonomously build a semantic object map over a long time span, focusing on small and medium-sized objects instead of furniture. Pangercic et al. [19] build a semantic map based on the detection of furniture parts in a set of 3D point clouds. Similar to our approach, they employ a description logic ontology as part of their system; however, their approach is strictly bottom-up (recognized objects are fed into the ontological knowledge base), whereas our approach uses semantic knowledge inside the recognition loop. Neumann and Möller [20] investigate the use of description logics for high-level scene interpretation tasks; they do not explicitly refer to semantic mapping, but their abductive approach to object and object aggregate perception is clearly suitable for the task.

CAD models have been used for object recognition before. For mass-produced objects, ranging from furniture over household appliances to tableware, CAD models exist and are widely available – either as the exact model directly from the manufacturer or as a similar model via sources like the Google 3D Warehouse. The first approaches in this direction started in the mid-nineties using vision based sensors [21] or a combination of a laser projector, a stereo camera and several additional cameras [22]. These approaches have in common that they try to recognize objects at a known position, i.e., the object in question is already centered in the obtained sensor data and no additional objects or occlusions are present in the sensor data. A more recent approach on a larger scale is presented by Bosché [23], where several matched 3D laser scans of a construction site are compared to a model in order to track progress and detect divergences between model and actual construction site. A prerequisite for this work is a correct model in advance, i.e., it is known what the sensors are supposed to measure and subsequently a quantitative analysis concerning the differences between expected measurements and received measurements is performed. In contrast to these approaches we neither demand a complete model of the perceived scene nor do we require the exact poses of candidates for object recognition in advance, but employ general domain knowledge to generate object hypotheses and use CAD models to refine these. From this characterization, our approach is similar to the work of Klank et al. [24], Mozos et al. [25], Usenko et al. [26] and Wohlkinger et al. [27]. The major difference to the system presented by Klank et al. [24] is that we perform the actual CAD matching with the 3D environmental data instead of 2D image data. Mozos et al. [25] and Usenko et al. [26] use CAD models to create synthetic point clouds of several types of furniture and extract geometric features which are used to build a vocabulary of objects via machine learning techniques. After a probabilistic Hough voting step to generate likely object hypotheses, they apply a RANSAC approach to fit the objects into the scene and confirm / refute their hypotheses. Like our approach, they use a parts-based representation of the furniture, and describe each part using geometric features. The features used in their approach are all directly based on the points belonging to each part. Our system includes a surface reconstruction step, which enables us to use the area of a planar patch as a feature. Another difference is that our system generates hypotheses using an ontological reasoner instead of probabilistic Hough voting.

An impressive recent approach to the problem of 3D object mapping is the SLAM++ system by Salas-Moreno et al. [28], which performs real-time recognition of 3D furniture models in RGB-D data. This allows them to build a 3D object map of the scene and use that map in the SLAM loop to simultaneously track the camera pose. The key difference to our approach is that our system is applicable to arbitrary 3D data like point clouds from laser scanners and does not rely on the special structure of RGB-D frames. This allows us to use arbitrary 3D sensors or use several registered frames as input to cope with the recognition of large objects that can not be captured in a single frame.

Lastly, the work by Lai et al. [29] should be recognized, which reports promising results in object labeling using full RGB-D information (color and depth). However their work focuses on streams of registered point clouds and thus the available data is usually much more complete than from a single frame.

Inspired by Gibson's [30] debated theory in psychology to understand perception as *direct* perception of affordances for an agent in its environment, several authors in computer vision and robotics have pursued approaches sharing the general structure of model-based perception – just that their models are formulated in terms of (directly perceivable) affordances rather than geometric features. Chemero and Turvey [31] give an overview about usages of the affordance concept in Robotics and AI, emphasizing the difference between traditional Gibsonian (direct perception related) and representationalist approaches co-existing in the literature.

3. Model-Based Object Recognition

In recent years, CAD models of many kinds of objects have become widely available. One resource of CAD models is Google's 3D Warehouse, which allows querying and retrieving CAD models of virtually any kind of object via the web. In the domain of furniture recognition, CAD models are often available directly from the manufacturer or from companies specialized in creating CAD models for interior designers. We use a database of CAD models supplied by our university's furniture manufacturer.

In this paper, we focus on the domain of furniture recognition for several reasons: First, due to the widespread use of CAD models in interior design, the availability of CAD models in this domain is especially strong. Second, most kinds of furniture feature a set of planar surfaces which can be robustly recognized in 3D point clouds. Third, due to the rigidness of furniture, these planar surfaces are in a clearly defined relation to each other.

Fig. 1 shows the embedding of our system in a general semantic mapping framework. We see our model-based object recognition method as complementary to appearance-based methods based on 2D image features or 3D shape features.

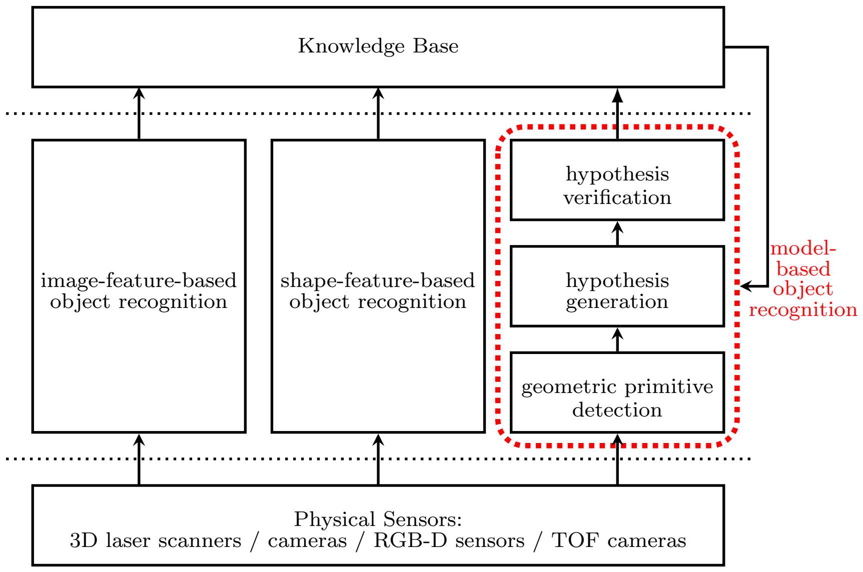
Knowledge Base
image-feature-based object recognition
shape-feature-based object recognition
hypothesis verification
hypothesis generation
geometric primitive detection
model-based object recognition
Physical Sensors: 3D laser scanners / cameras / RGB-D sensors / TOF cameras

Figure 1: System overview. While the present paper is focused on model-based object recognition, we consider this method as yielding complementary information to standard recognition methods. So in a more general system architecture, they may well co-exist. We will not deepen this issue here (Figure reproduced from [15]).

Using the information contained in CAD models for object recognition has several advantages. Instead of having one classifier for each kind of object, only the geometric primitives have to be detected. Based on these, objects are reconstructed. Also, no classifier training and no labeled training sets are required; to add a new object class, only the CAD model is required. In the future, it would even be conceivable that such a CAD model could be retrieved on-line from the web. Another advantage is that once an object is recognized, the corresponding part of the sensor data can be replaced by the CAD model, thus filling up occlusions in the sensor data.

On the other hand, appearance-based methods have an advantage where the to-be-recognized object is non-rigid, does not consist of clearly identifiable geometric primitives of a certain minimum size or where labeled training data, but no CAD model is available.

We see our model-based object recognition method as an instance of a more general system architecture (Fig. 1), consisting of three steps: (1) surface reconstruction (Sec. 3.1), as an instance of geometric primitive detection, transforms the input point cloud into a triangle mesh and extracts planar regions; (2) planar region classification (Sec. 3.2), as an instance of hypothesis generation, classifies the planar regions, detects furniture objects, and calculates initial pose estimates based on the planar regions; (3) final pose adjustment (Sec. 3.3), as an instance of hypothesis verification, computes the final pose using ICP, and places the corresponding CAD model in the scene.

3.1. Surface Reconstruction

Surface reconstruction is our implementation of the geometric primitive detection step in Fig. 1. The plane extraction for object recognition is done on a mesh representation of the surfaces captured with the Kinect camera using the Las Vegas Surface Reconstruction Toolkit (LVR) [32]. LVR provides an open source C++ library with implementations of several algorithms for polygonal map generation. A comparison with state of the art reconstruction algorithms [33] showed that LVR's Marching Cubes implementation outperforms other state of the art methods like Poisson Reconstruction [34] or Delaunay based methods [35, 36] in arbitrary environments with respect to geometrical accuracy and topological soundness of the generated meshes. The surface reconstruction process mainly consists of two steps: initial mesh generation using Marching Cubes followed by a post-processing pipeline of several mesh optimization and segmentation algorithms.

Mesh generation is done using an optimized Marching Cubes implementation that utilizes Hoppe's distance function [37] to estimate an isosurface representation of the point cloud data. To generate this isosurface, surface normals for the data points have to be estimated. This is done using an adaptive RANSAC-based approach that is optimized for sparse data sets containing Kinect specific discretization effects and noise [32]. To estimate a normal, a local plane is fitted to the k nearest points of a query point ("k-neighborhood") using RANSAC. For performance reasons, the k-neighborhood should be as small as possible. Unfortunately, the point density of Kinect frames and laser scans is not constant, so the k-value has to be adapted to ensure stable results. Due to discretization effects of the measurement principle of the Kinect, the point clouds often contain line shaped artifacts. If the k value is too small, the k-neighborhood will be aligned on such a line, which makes the normal approximation depend solely on local noise. To avoid this effect, we analyze the bounding box of the k-neighborhood. In line shaped alignments, one side of the bounding box will be significantly longer than the others. If such a condition is detected, the

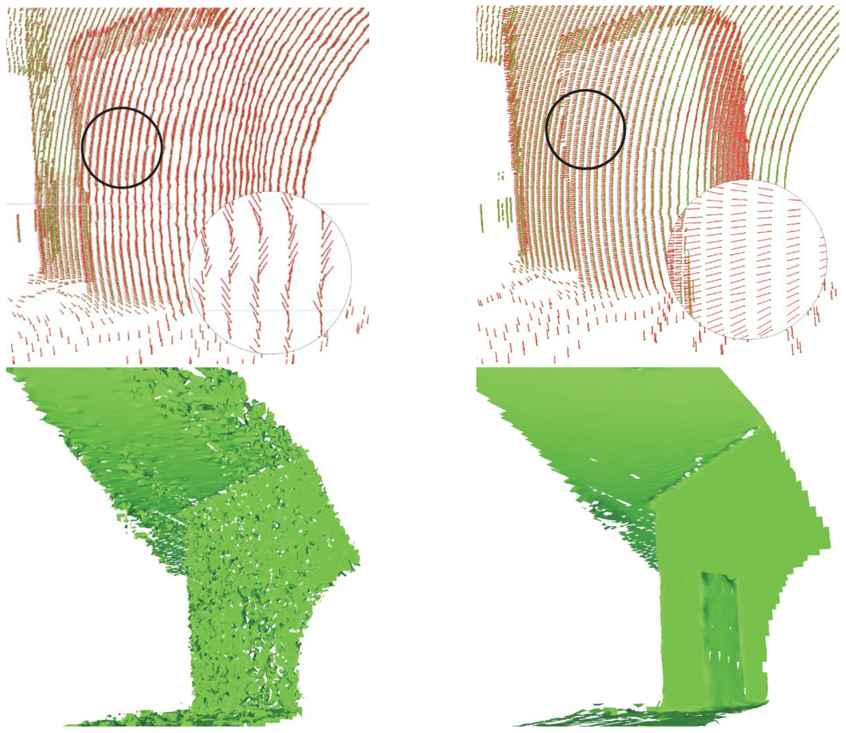

Figure 2: Normal estimation in noisy point cloud data. In sparse data sets, the local configuration of a fixed k neighborhood might be unsuitable for normal estimation (left). With adaptive growing, we are able to produce accurate results for the estimated normals, which results in connected surface reconstructions (right).

initial k value will be increased dynamically until the bounding box criterion is fulfilled. An example for this process is shown in Fig. 2. The combination of RANSAC based normal estimation with k adaption ensures stable normals even in sparse and noisy point clouds and is thus well suited for Kinect input data. Usually a k-neighborhood of 10 neighbors is sufficient.

To approximate a triangle mesh representation to the isosurface defined by the points and normals, a modified Marching Cubes algorithm is used. This initial mesh is enhanced by several mesh optimization algorithms. To remove artifacts based on outliers in the point cloud data, small clusters of connected triangles are automatically removed. Furthermore, LVR delivers a hole filling procedure to close holes in the mesh that result from occlusions. After initial mesh generation, hole filling and outlier removal, connected planar patches in the mesh are extracted. The planar segmentation is done using a region growing approach. For an arbitrarily chosen start triangle, the normals of all neighboring triangles are analyzed. As long as the normal of a neighbor triangle differs no more than a user defined threshold from the start triangle, this face is marked as used and a new search is started recursively. In our implementation, we used a threshold of $3°$ to account for the present sensor noise. Initial experiments showed that higher tolerances lead to under-segmentation, while a stronger threshold will abort recursion due to fluctuations resulting from sensor noise. All patches on the same plane are stored in a list that represents the current plane. The recursive search is carried on until a bend in the surfaces or a border edge in the mesh is found. After all triangles of a planar region are found, a new search is started from an unused triangle. This process is carried on until all triangles in the mesh have been checked (cf. Algorithm 1).

The output of this algorithm is a set of planar clusters represented through contour polygons that can be used to generate the model hypotheses for the object recognition process. In principle, other approaches to detect planar polygons like [38], [39] or [40] could be used. The main benefit of our approach is that we can extract the exact concave contours of the planar regions, while the other approaches are using unions of convex hulls or α-shapes.

To compute model hypotheses, the actual size of an extracted plane is needed. In contrast to point clouds, the exact area can be easily computed in a mesh representation by summing up the areas of the created triangles. In real-life application scenarios, the interesting surfaces of furniture will usually be populated with objects on top of them, especially in table top scenarios. For example, take a laid breakfast table where the table top contains dishes,

Algorithm 1 The mesh simplification algorithm. Faces that have similar surface normals are detected. The border edges of these planar areas are fused to polygons.

```
function RegionGrowing
    for all faces do
        Fuse(current normal, current face, current list)
        border list ← current list
        CreatePolygon(border list)
        current list ← empty
    end for
end function

function Fuse(start normal, current face, list of borders)
    current face ← visited
    for all unvisited neighbors of current face do
        angle ← start normal · neighbor normal
        if angle < $\epsilon$ then
            Fuse(start normal, neighbor, list of borders)
        else
            list of borders ← border edge to neighbor
        end if
    end for
end function
```

bowls and various other objects that are needed for a proper breakfast. These objects will certainly create occlusions and holes in the reconstruction of the table top plane. As long as our region growing algorithm can find connected patches of the surface between the present objects, we will get a representation of that area, but the estimated area will be significantly reduced due to the occlusions. To restore the initial plane, we re-triangulate the outer contour of the plane, which is detected via topological sorting. To get an optimized contour representation, we fuse edges on the same line via contour tracing using the `psimpl` library [41]. The outer contour is then triangulated using the OpenGL tesselator. This way, all holes within the plane are closed and we get a realistic approximation of the area of an occluded planar surface (cf. Fig. 4).

This approach works fine as long as the outer contour is not interrupted by shadows of present objects. In this case, the shadow might break the outline and create a bay in the contour which in turn will decrease the estimated surface. Alternatively, one could use a convex hull approach to estimate a surface, but by doing so the surface of non-convex polygons – like the L-shaped desk in the office dataset (Sec. 4) – would be overestimated. Our approach can be used for arbitrary shapes. Another problem occurs when a surface is too populated. In this case, the region growing procedure will not be able to find a connected remaining surface and the estimated plane will be split. An approach to solve this problem is to detect close patches which satisfy similar plane equations. In these cases, their areas can be summed up. While these optimizations make the area estimation more stable, the classification step that is described in the next subsection still needs to be robust to variations from the true area size. An analysis to evaluate the robustness of our approach is presented in Section 4.

To determine whether a plane is horizontal or vertical, we analyze the orientation of the normal of a planar patch. If the angle between the normal and the y axis is smaller than $3°$, we label it as horizontal. To classify vertical patches we project the normal onto the x-z plane. If a patch is perfectly perpendicular to the x-z plane, the length l of the projection is exactly 1. To account for small orientation errors, we classify all patches with $\|l-1\| < 0.3$ as vertical.

3.2. Planar Region Classification

Planar region classification is our implementation of the hypothesis generation step in Fig. 1. Once all planar regions have been extracted in the

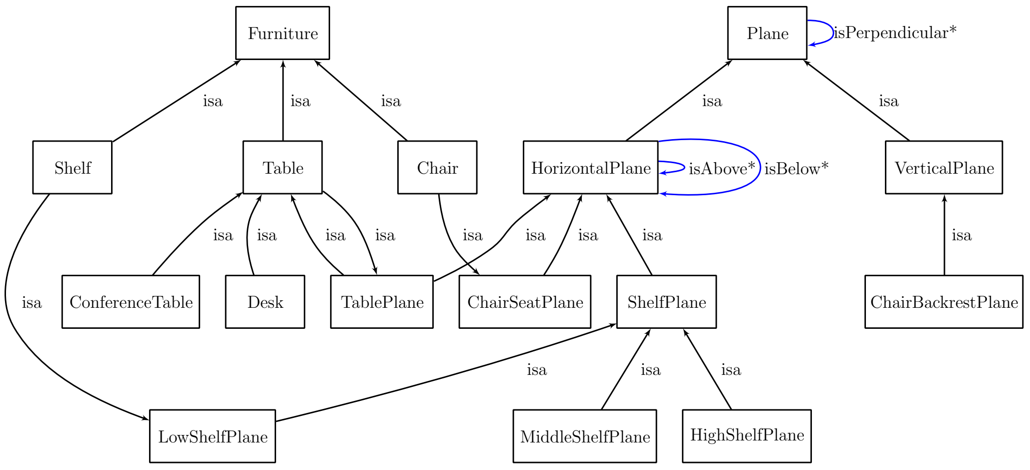

Figure 3: Parts of the OWL-DL ontology used for classification: classes (black) and properties (blue). (Figure reproduced from [16])

previous step, those regions corresponding to pieces of furniture have to be classified. Here, we make use of the fact that most pieces of furniture are comprised of planar structures that have a certain size, orientation, height above ground and spatial relation to each other. These features (and combinations of features) are expressed in an OWL-DL ontology in combination with SWRL rules.

The Web Ontology Language (OWL) is the standard proposed by the W3C consortium as the knowledge representation formalism for the Semantic Web. One of its three sub-languages, OWL-DL, corresponds to a Description Logic [11], a subset of First-Order Logic that provides many expressive language features while guaranteeing decidability. It has been extended by SWRL, the Semantic Web Rule Language, [42], which allows to write Horn-like rules in combination with an OWL-DL knowledge base and includes so-called built-ins for arithmetic comparisons and calculations. We decided to use OWL-DL as the knowledge representation format for this work for several reasons: OWL-DL ontologies can be easily re-used and linked with other sources of domain knowledge from the Semantic Web, they easily scale to arbitrarily large knowledge bases, and fast reasoning support is available. In our implementation, we use the open-source OWL-DL reasoner Pellet [43], which provides full support for OWL-DL ontologies using SWRL rules.

The class hierarchy of the ontology we use here is shown in Fig. 3. The basic classes are Furniture (the parent class of all recognized furniture objects) and Plane (the planar regions of which Furniture objects are comprised). A set of SWRL rules is applied to the extracted planar regions to assign them classes in the Plane sub-hierarchy; for example, the lower plane of a shelf can be characterized by the following SWRL rule:

```
LowShelfPlane(?p) ← HorizontalPlane(?p)
   ∧ hasSize(?p, ?s) ∧ swrlb:greaterThan(?s, 0.01)
   ∧ swrlb:lessThan(?s, 0.5) ∧ hasPosY(?p, ?h)
   ∧ swrlb:greaterThan(?h, 0.08)
   ∧ swrlb:lessThan(?h, 0.18)
```

These rules are not exclusive, so one planar region can receive multiple labels (e.g., MiddleShelfPlane and ChairSeatPlane). The definitions of the classes in the Furniture sub-hierarchy refer to these labels; e.g., the fact that a Shelf consists of three planes on top of each other can be stated as:

```
Shelf ≡ LowShelfPlane and
   (isBelow some (MiddleShelfPlane and
      (isBelow some HighShelfPlane)))
```

Likewise, chairs are defined by a seat and a backrest, both with certain sizes, orientations, heights above ground and which are perpendicular to each other. In this work, these rules (which encode the structural model of the object) were constructed manually. The parameters can be measured directly from the CAD model – e.g., in the example above, the lower shelf plane has a height above ground of 0.13 m; adding a margin of 0.05 m to compensate for errors in the estimated height, one arrives at the specified range (0.08 m; 0.18 m). We hope to automate this process in future work.

The classification of planar patches into furniture objects is performed by the Pellet reasoner. Initially, each planar patch that was extracted during surface reconstruction, along with its geometric features (size, position, orientation, bounding box) and relations to other planar patches is added as an individual to the ontology's ABox. Next, the reasoner jointly classifies all planar patches, using the SWRL rules and class definitions outlined above.

For each detected object, initial position and orientation are estimated. The position is always the centroid of their main plane. Since chairs have two perpendicular planes (the backrest and the seat), a unique orientation can be calculated by the vertical component of the difference vector between the centroids of those planes (assuming an upright position of the chair). For tables and shelves, the PCA of the points corresponding to their main plane is calculated to define the orientation. Note that the PCA of a rotationally symmetric object, such as a round table, is not well-defined. This does not impact the recognition rate of our system, only the initial orientation estimate; if the object has distinct non-planar features (such as table legs), this orientation error can be corrected by the next processing step.

3.3. Final Pose Adjustment and Model Replacement

Final Pose Adjustment is our implementation of the hypothesis verification step in Fig. 1. The initial pose estimate calculated in the previous step is potentially inaccurate, since it wholly depends on an abstraction of the recognized objects as sets of planar regions. Solving this problem requires closing the loop from the output of our reasoning process back to the low-level sensor data.

To improve the pose estimate, we match a CAD model with the original point cloud data using the well-known Iterative Closest Point (ICP) algorithm [44]. Since ICP needs two point clouds as input, we create a surface sampling of the CAD model to fulfill the algorithm's requirements. The sampling process assumes that the surface of the CAD model is represented by a collection of triangles, which is common for standard CAD formats. Each triangle's area is sampled by 3D points, according to a desired point density, which should correlate with the measurement point density of the used sensor. Each triangle can be sampled either in a regular fashion (see [45] for details) or in a random fashion, where the number of random sample points is determined by the size of the triangle's area and the desired point density. Since we sample the complete surface of the CAD model, the resultant point sampling does not consider (self-)occlusion from the sensor's point of view. While this is generally a valid idea when matching against a registered point cloud (combining data from several individual frames), there certainly is room for improvement concerning the matching process for single frames. During the matching process, all sampled points are assigned the same weight, which in general yields stable and satisfactory results (see experiment 4.2). Different weights for certain points of the surface sampling, e.g., points belonging to the seating surface and backrest of a chair might improve the alignment of the CAD models. We have not investigated this idea yet, but it remains an interesting open point for further research. While the output of ICP, i.e., the average point-to-point error, gives us a rough idea how well the sampled CAD model fits to the point cloud data, a more sophisticated evaluation of the final result is an important part of future work.

4. Results

We performed three experiments to evaluate the robustness and accuracy of our recognition system. First, we investigated the effectiveness of our hole filling procedure separately to see whether it is capable of estimating the true surface area of a table under the influence of increasing amounts of clutter. Second, we tested the robustness of our system with respect to the required similarity between CAD model and actual object. Lastly, we evaluated the detection accuracy of our complete system on two series of point clouds captured by a mobile robot.

4.1. Robustness Against Occlusion

In real life applications, furniture is usually used to store objects, so an obvious problem for our detection procedure is that the surfaces relevant for recognition may be partially occluded. To evaluate the robustness of the surface extraction procedure against occlusions, we gradually added typical objects like books, cups and bottles to a table surface and tried to segment the table top. We compared the table top surface that was determined by summing up the reconstructed triangle surfaces during simple region growing with the contour triangulation approach. The results of this experiment are shown in Fig. 4 and Table 1. Using region growing, the shadows caused by the present objects reduce the detectable surface area with increasing number of objects. These holes are filled up when the outer contour is triangulated. Hence, the detected area for contour triangulation stays close to ground truth until the outer contour is broken by a shadow (here in the presence of 12 objects), while the detected area using region growing gradually becomes smaller due to shadows. The small variances in the reconstructed areas are caused by noise in the input data that can lead to slightly different triangulations using LVR's Planar Marching Cubes algorithm, which shifts the vertices of the contours to the nearest data point [46]. Since the reconstruction is based on the noisy input, it is unlikely that the exact ground truth value is hit.

4.2. Robustness of CAD Matching

While various CAD models for a wide range of objects, including furniture, are freely available on the world web wide via sources like Google 3D

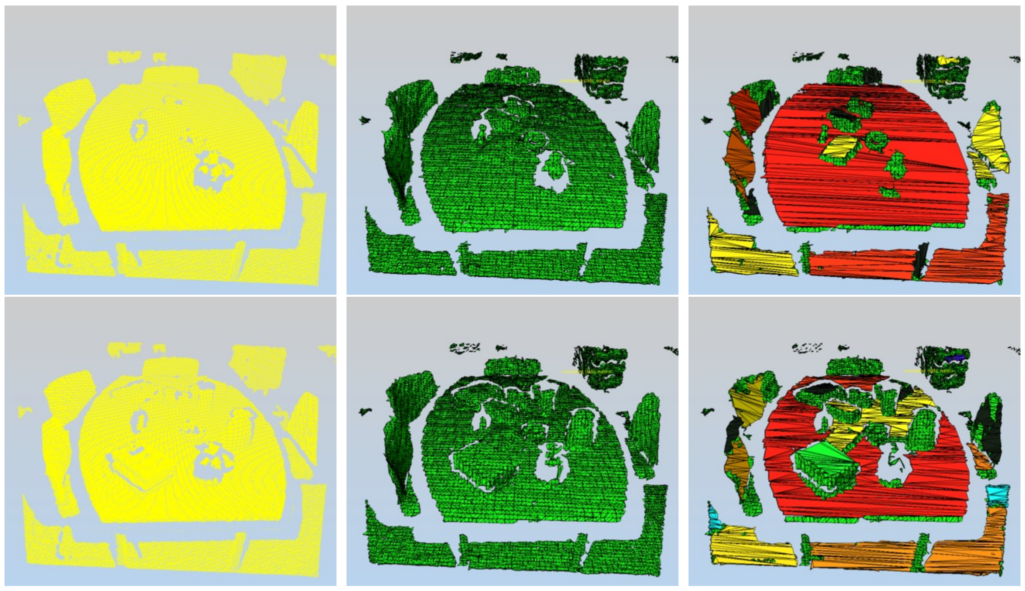
Figure 4: Segmentation results for a table top setup. First column: the captured point clouds; second column: initially created triangle mesh; third column: segmentation results. Triangles that were not classified as belonging to a planar patch are rendered in green. In each step more objects were added. In the last line a shadow disrupted the outer contour and the segmentation broke the table top plane into two clusters. (Figure reproduced from [16])

Warehouse, it is often not easy to find the exact model for a particular piece of furniture. Also, the physical object might differ from the CAD model for other reasons (e.g., damage or other modifications).

To investigate how robust our ICP matching step is with respect to such differences between CAD model and actual object, we conducted an experiment where we matched several CAD models of chairs against recorded point data of a chair. A photograph of the chair and a view of the resulting point cloud can be seen in Fig. 5. The point cloud displayed in Fig. 5b was created from five registered Kinect frames.

We matched this data against six different CAD models of chairs that were retrieved from Google 3D Warehouse (Fig. 6). We considered the Chair 1 model (top left) as a best fit of the actual chair, while Chair 2–4 were expected to be similar enough to produce meaningful matching results. For comparison, we included a model of a stool (Chair 5) and a wing chair (Chair 6), which we expected to be too different from the chair in the sensor data

Table 1: Reconstructed areas in the table top experiment under increasing amounts of clutter (reconstructed table top area in m^2 and as percentage of ground truth). First row: only region growing. Second row: region growing, followed by contour triangulation.

	0 obj.	2 obj.	3 obj.	4 obj.	5 obj.	6 obj.	7 obj.	12 obj.
Region Growing	1.50 93%	1.47 92%	1.47 92%	1.40 87%	1.35 84%	1.26 79%	1.17 73%	0.95 59%
Contour Triangulation	1.50 93%	1.50 93%	1.49 93%	1.52 95%	1.52 95%	1.52 95%	1.50 93%	1.20 75%

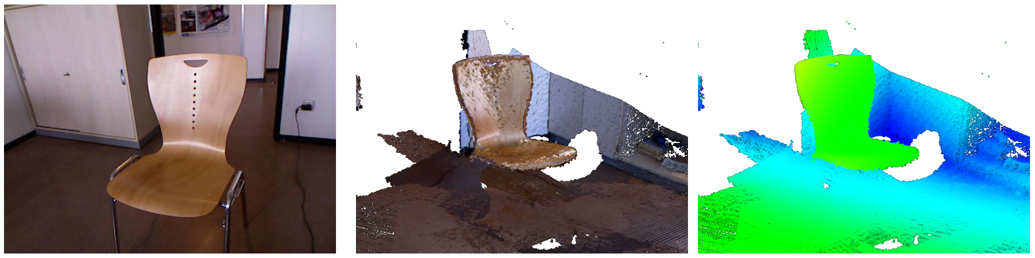
(a) (b) (c)

Figure 5: Registered point cloud input data for experiment 4.2: (a) photograph of the chair; (b) corresponding point cloud composed of 5 Kinect frames; (c) point cloud. Points are colored according to their distance to the z axis. Note that the Kinect did not provide sensor readings for the chair legs due to their highly specular surface (evident in (c)).

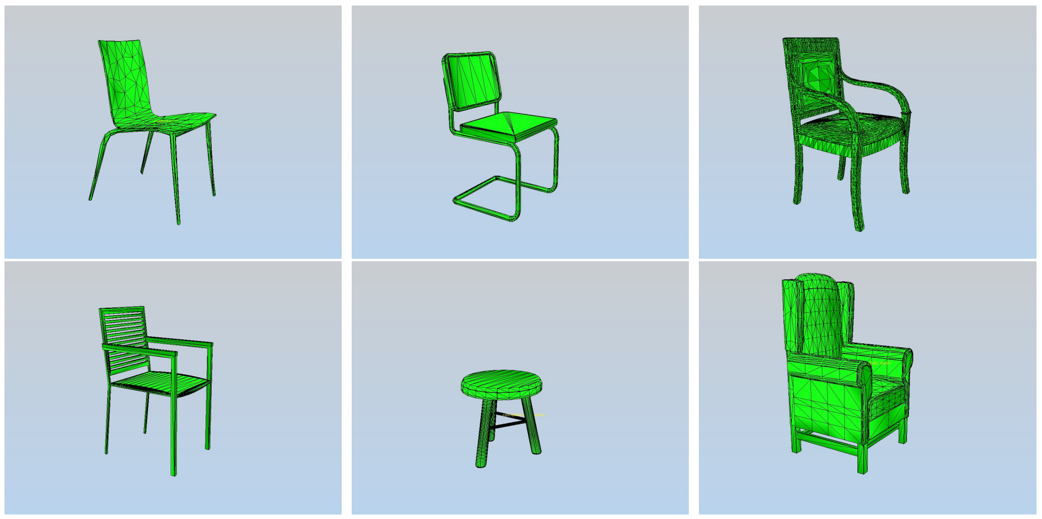

Figure 6: CAD models of chairs used for matching against chair depicted in Fig 5. We considered the top left model to be a best fit, while apart from the stool and the wing chair all models are similar enough in appearance to provide meaningful results.

to produce good results.

To assess the performance, we define a *reference pose* $\mathbf{X}_{i,\mathrm{ref}}$ as a replacement for ground truth which is not available. The reference pose was calculated for each model by manually aligning the model to the point cloud, followed by ICP for the final adjustment (Fig. 8, top row). All models ended up close to the initial pose except for Chair 6 (the wing chair); this is due to the large number of points in the wing chair's base.

For the actual experiment, our ICP model alignment step was run on each chair model from three different initial poses, resulting in the *final pose* $\mathbf{X}_i$; see Fig. 7 for the initial poses and Fig. 8 for the final poses.

A pose $\mathbf{X}_i$ itself is composed as a 2-tuple $\left(\hat{\mathbf{X}}_i, \tilde{\mathbf{X}}_i\right)$, where $\hat{\mathbf{X}}_i = (x, y, z)^T$ describes the translation part of the pose and $\tilde{\mathbf{X}}_i = (p, q, r, s)^T$ denotes the rotation as a quaternion. In order to provide a meaningful error between a reference pose $\mathbf{X}_{i,\mathrm{ref}}$ and final pose $\mathbf{X}_i$, we calculate the translational error $e_{\mathrm{translation}}$ and the rotational error e_{rotation}. The translational error is simply defined as the Euclidean distance between two poses:

$$e_{\mathrm{translation}} = \|\hat{\mathbf{X}}_i - \hat{\mathbf{X}}_{i,\mathrm{ref}}\| \tag{1}$$

while $e_{\mathrm{rotation}}$is a unit quaternion distance metric introduced by Kuffner [47]

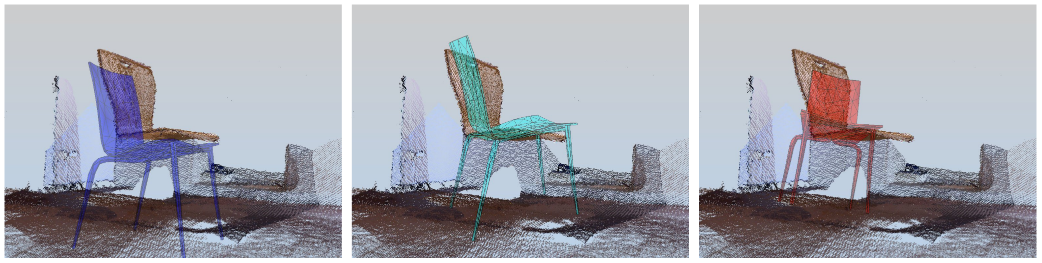

Figure 7: Initial pose visualization, depicted by the example of the reference CAD model (Chair 1). Initial translation and rotation errors ($e_{\text{translation}}/e_{\text{rotation}}$) are as follows. Pose 1 (left): 13.87 cm/0.13°; Pose 2 (middle): 0.65 cm/27.50°; Pose 3 (right): 36.34 cm/22.5°.

Table 2: CAD matching results. †: Note that this model is rotation invariant around one axis, so the rotation error does not necessarily reflect the actual quality of the final pose.

	final pose error					
	pose 1		pose 2		pose 3	
	$e_{\text{translation}}$	e_{rotation}	$e_{\text{translation}}$	e_{rotation}	$e_{\text{translation}}$	e_{rotation}
chair 1	0.5 cm	0.47°	0.5 cm	0.48°	0.5 cm	0.0°
chair 2	0.0 cm	0.1°	0.1 cm	0.11°	0.0 cm	0.0°
chair 3	0.0 cm	0.04°	0.0 cm	0.04°	0.0 cm	0.01°
chair 4	0.1 cm	0.04°	0.1 cm	0.05°	1.9 cm	34.07°
chair 5 †	3.1 cm	12.22°	2.2 cm	22.55°	3.0 cm	12.77°
chair 6	29.5 cm	3.56°	11.1 cm	42.81°	10.9 cm	42.85°

as

$$e_{\text{rotation}} = \arccos |\tilde{\mathbf{X}}_i \cdot \tilde{\mathbf{X}}_{i,\text{ref}}| \tag{2}$$

Using a unit quaternion distance metric has the advantage that it does not suffer from ambiguities, unlike the common method of comparing Euler angles.

The resulting final errors for the six CAD models in this experiment are shown in Table 2. As expected, Chair 1–4 converged about equally well to the reference pose (except for one outlier from Chair 4), whereas Chair 5 and 6 did not converge well.

However it has to be noticed that the data used in this experiment was not very challenging, since the chair is completely captured in the point cloud

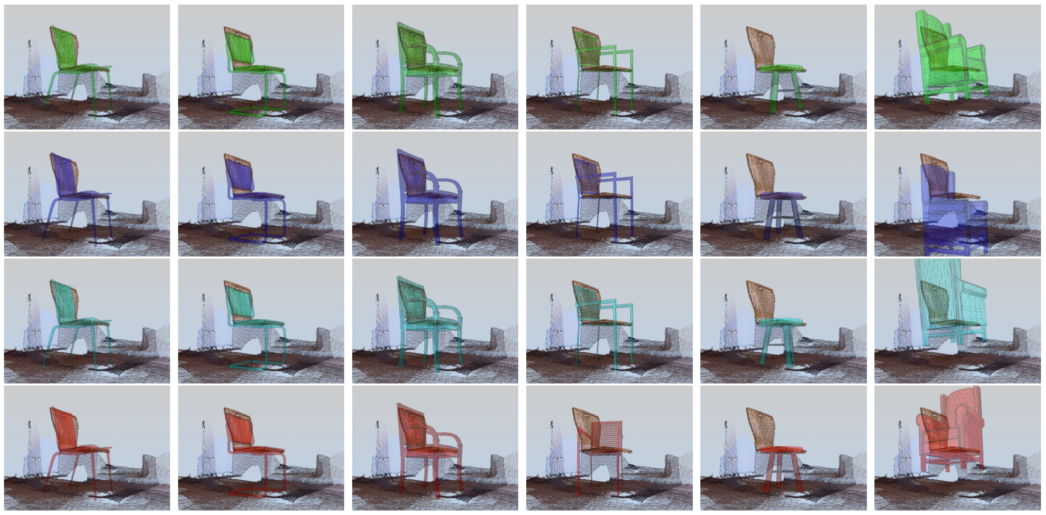

Figure 8: Transparent overlay of the final poses obtained by CAD matching with the point cloud. Top row: results from an aligned pose estimation. Second from top: results from pose estimation below actual pose (Pose 1). Second from bottom: results for slight displacement and rotation error (Pose 2). Bottom row: results for larger displacement and rotation error (Pose 3). See Fig. 7 for visualization of initial pose.

data and there are no other objects near the chair. To evaluate the effects of choosing a different CAD model on the performance of the complete system, we ran the complete pipeline as explained in the next subsection once for each chair model. Table 3 compares the final translation and rotation errors after ICP alignment on the seminar room dataset (see Sec. 4.3, Table 4c). The results clearly indicate that as long as the CAD model is "similar enough" (chairs 1–4) to the objects found in the data, our system works. The results for chairs 2–4 were even slightly better than for chair 1, which we considered the best fit for the actual object and which is used in the subsequent experiments.

4.3. Complete System

To evaluate our recognition system, we captured two series of point clouds from a Kinect camera mounted on a mobile robot (see Fig. 9). In the first scenario, the robot was tele-operated around an office while continuously capturing point cloud data at 2.2 Hz, resulting in a total of 431 point clouds. The office contained 13 recognizable objects from 5 classes (1 desk, 1 conference table, 1 office chair, 5 conference chairs and 5 book shelves). For the

Table 3: Final translation and rotation errors (after ICP) for the different chair models, compared on the complete seminar room dataset (single point clouds). Results are averaged on all 358 true positive detections of chairs in the dataset. The average translational/rotational errors of the initial pose estimate are 7.3 cm/20.3°. The pose errors are expressed according to equations (1) and (2); †: Note that this CAD model is rotation invariant around one axis, so that the rotation error does not necessarily reflect the actual quality of the final pose.

	$e_{\text{translation}}$	e_{rotation}
Chair 1	8.3 cm	10.0°
Chair 2	6.0 cm	7.8°
Chair 3	7.3 cm	7.5°
Chair 4	7.9 cm	9.3°
Chair 5[†]	10.9 cm	23.7°
Chair 6	22.6 cm	16.8°

second dataset, the robot was driven through a seminar room, capturing a total of 379 point clouds. The objects present in this dataset are 12 seminar tables and 20 chairs. One challenging aspect of this dataset is that there is a high level of occlusion.

For both datasets, we registered the point clouds into a consistent full-scene point cloud, using SLAM6D from 3DTK [48]. The full-scene point clouds were used to generate the ground truth poses for each piece of furniture by hand. These poses are used to evaluate the results of our classification and the ICP refinement step.

Ground truth data for each frame was generated by manually labeling each frame with the information which of the objects occur in that frame. The ground truth poses of each object were estimated by manually placing each CAD model into a scene consisting of the fully registered datasets. In combination with the camera trajectory obtained from the registration process, the ground truth object poses in each frame can be computed. A detection is considered "true positive" if its dis-

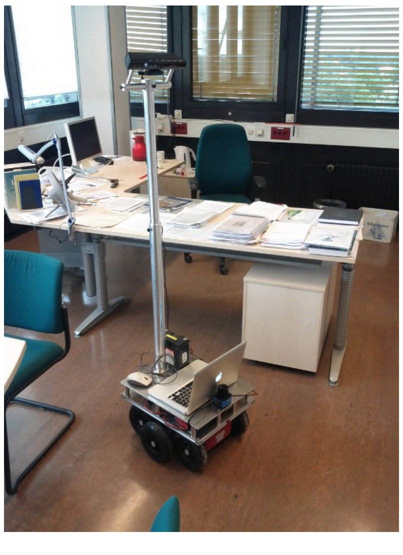
Figure 9: The Kurt robot used for our experiments. (Figure reproduced from [16])

tance to the nearest true object pose of the same class was below a threshold, depending on the CAD model's size (15 cm for chairs, 25 cm for shelves, 45 cm for tables). If multiple detections fell into that range, only the nearest was counted as a true positive, and the others as false positives.

These high thresholds were chosen based on experience and reflect the diminishing depth resolution of the Kinect sensor (at 5 m the depth resolution is approx. 5 cm – even without noise depth measurements can differ substantially from the real distance) accompanied by additional random noise, as well as the size of the discriminating planar surfaces of the object classes. Furthermore if all individual point clouds are registered perfectly, there will still be noisy "shadow points" around each object. Our ground truth poses are located in the middle of the noisy point cloud resembling the objects in question.

Note that the recognition system itself does not require prior registration; it can work both directly on unregistered single-frame point clouds or on a registered full-scene point cloud. Both approaches have their advantages; directly processing each frame eliminates the computational cost for registration, removes the risk of registration failures, and avoids artifacts arising from combining many point clouds from a noisy sensor. Single frame processing is therefore better-suited for online operation in an incremental semantic mapping framework. On the other hand, the narrow field of view and occlusions – that may be recovered by viewing at an object from different angles – make it more likely that a piece of furniture is not fully visible.

The detection results both for single frames and the full scene on both datasets are shown in Table 4. In addition to the detection results, the translation and rotation error of the initial guess (based on PCA for tables and shelves, and based on the vector from backrest to seat for chairs) and the translation and rotation error after ICP pose correction are shown. Fig. 11 depicts some exemplary object detections, Fig. 10 shows the final results of the system running in full-scene mode.

For the single point clouds, our approach achieves detection rates of 46.0 % and 79.4 % on the two data sets. We expect that these results can be improved significantly in the future by integrating information over several frames instead of treating each frame independently. The results show that our approach is not only robust enough to deal with the noise present in low cost 3D sensors, but also copes with occlusion and partial visibility, typical for sensors with a small opening angle. Unexpectedly, the final ICP pose correction did not improve the average initial guess on single frames for the first dataset. We attribute this to the fact that we matched a complete CAD model to a partially occluded object view; possible solutions are outlined in the next section. In the second dataset however, ICP clearly improved the rotation of the chairs compared to the initial guess from the SWRL rules.

The results also show varying detection success for the different classes. Shelves have one of the lowest detection rates and highest final pose error in our experiments. This is not surprising: All shelves in our data set were completely filled with books, so only a small portion of the actual shelf was visible. This also explains why the final pose correction performed worse on shelves compared to the other object classes. In addition, we currently do not handle aggregates of objects: If two shelf segments or two tables are positioned with no gap between them, the planar classification will combine them into one potential object, with the possible location at the center of the combined plane. This usually leads to one false positive (for the non-existent object at the center location) and several false negatives for actual objects creating the aggregated plane. Another problematic object is the big L-shaped desk: The desk is so big that only a small part of it is visible in most single frames.

A comparison between single-frame and full-scene mode reveals that both approaches have their complementary strengths and weaknesses. Both classification accuracy and final pose error for most objects (especially big ones, like tables) are better in full-scene mode, since there are less problems with partial visibility due to occlusion or limited aperture. On the other hand, the detection rate for chairs in the seminar table dataset is higher in single-frame mode. The main reason for this seems to be that since chairs are relatively small compared to tables, limited aperture doesn't pose as much of a problem in single-frame mode. On the other hand, since the chairs were relatively close together, registration errors and accumulated sensor noise in full-scene mode often lead to two chairs being recognized as one object, preventing detection.

4.4. Runtime of the system

The runtime performance of our system and its components are shown in Table 5. All experiments have been performed on a standard laptop (2.6 GHz Core i7 CPU, 8 GB RAM). The single point clouds are processed at full

Table 4: Detection Rates. The pose errors are expressed according to equations (1) and (2).

	true pos.	**false pos.**	**false neg.**	**initial transl. error**	**initial rot. error**	**final transl. error**	**final rot. error**	**precis.**	**recall**	F_1 **score**
				(a) office dataset (single point clouds):						
Shelf	82	53	237	16.2 cm	5.1°	60.1 cm	27.2°	60.7 %	25.7 %	36.1 %
OfficeChair	8	19	6	5.8 cm	61.0°	6.0 cm	10.8°	29.6 %	57.1 %	39.0 %
ConfChair	100	190	114	9.0 cm	51.9°	11.0 cm	56.8°	34.5 %	46.7 %	39.7 %
Desk	10	11	29	31.6 cm	125.6°	53.7 cm	118.8°	47.6 %	25.6 %	33.3 %
ConfTable	81	0	0	8.5 cm	8.8°	9.4 cm	6.2°	100.0 %	100.0 %	100.0 %
total	281	273	386	11.7 cm	28.7°	26.2 cm	34.5°	50.7 %	42.1 %	46.0 %
				(b) office dataset (full registered scene):						
Shelf	1	0	4	24.7 cm	1.0°	131.6 cm	4.1°	100.0 %	20.0 %	33.3 %
OfficeChair	1	0	0	5.2 cm	68.1°	4.7 cm	14.2°	100.0 %	100.0 %	100.0 %
ConfChair	3	2	2	10.9 cm	55.9°	10.1 cm	49.1°	60.0 %	60.0 %	60.0 %
Desk	1	0	0	41.8 cm	21.2°	12.8 cm	6.7°	100.0 %	100.0 %	100.0 %
ConfTable	1	0	0	2.5 cm	4.3°	6.3 cm	0.6°	100.0 %	100.0 %	100.0 %
total	7	2	6	15.3 cm	37.5°	26.5 cm	24.7°	77.8 %	53.8 %	63.6 %
				(c) seminar room dataset (single point clouds):						
Chair	358	26	257	7.3 cm	20.3°	8.3 cm	10.0°	93.2 %	58.2 %	71.7 %
SemTable	522	153	21	9.3 cm	2.5°	9.2 cm	2.4°	77.3 %	96.1 %	85.7 %
total	880	179	278	8.5 cm	9.7°	8.8 cm	5.5°	83.1 %	76.0 %	79.4 %
				(d) seminar room dataset (full registered scene):						
Chair	6	2	14	5.7 cm	16.3°	6.0 cm	9.8°	75.0 %	30.0 %	42.9 %
SeminarTable	11	0	1	4.2 cm	1.2°	3.3 cm	1.2°	100.0 %	91.7 %	95.7 %
total	17	4	15	4.7 cm	6.537°	4.3 cm	4.252°	81.0 %	53.1 %	64.2 %

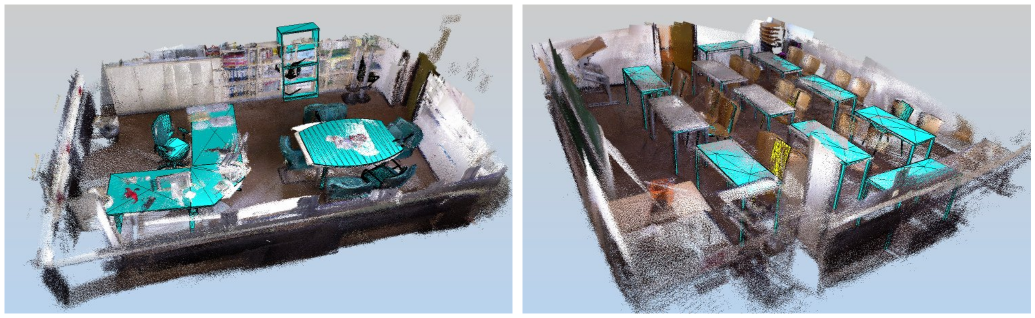
Figure 10: Final results of our system in full-scene mode (registered full-scene point clouds overlaid with the recognized furniture models). Left: Office dataset (431 frames). Right: seminar room dataset (379 frames). Note that point color information is displayed for reader convenience, our approach does not require it.

Table 5: Run times of the three steps in our processing pipeline on the seminar room data set. The results for single point clouds were averaged over all 379 point clouds.

	single point cloud	**full registered scene**
Surface Reconstruction	6.53 s	117.06 s
Planar Region Classification	1.02 s	2.67 s
Final Pose Adjustment	1.53 s	26.11 s
Total	9.09 s	145.84 s

resolution (245,304 points on average), while the full scene was downsampled to about one tenth of all points (9,475,220 points).

Most of the processing time is spent on the surface reconstruction step. This could in principle be replaced by faster but less accurate methods. For the RGB-D data used here, Kinect Fusion is a good candidate, although the meshes have to be post-processed to get a topologically correct mesh representation [33], which is required for region growing. For our experiments, we decided to use the Marching Cubes implementation from our LVR library, since it computes meshes that are accurate and topologically sound. The classification and pose adjustment steps are fast. The whole scene consisting of 379 point clouds, i.e., about 9.5 million points, was processed in about 2.5 Minutes.

5. Summary and future work

We have presented a semantic mapping system that creates a triangle mesh of an office environment, detects several classes of furniture, and replaces them by their corresponding CAD models, based on Kinect point cloud data captured using a mobile robot. The system was evaluated both on single frames and fully registered scenes of two datasets containing 810 single point clouds. Our system achieved a detection rate of 46.0 % for the office dataset, containing one particularly large object and several seamlessly connected instances of shelves and of 79.4 % on the seminar room with less variation of the classified furniture.

The current system creates a *hybrid* semantic map (i.e., the map data contains geometric information as well as semantic knowledge); it does so *incrementally* by processing each RGB-D frame separately; and it also does so in *closed loop* by feeding the output of the reasoning system back into the low-level data interpretation routine. In future work, we intend to implement an active data interpretation / gathering loop that identifies and actively explores regions with potentially valuable but missing data, thereby closing the loop even down to the action level, not just the perception level.

Another necessary improvement is a matching criterion that decides how well a CAD model was matched in the point cloud. Such a criterion could be used to reject false hypotheses, and to disambiguate between similar models.

The performance of the final ICP pose correction could be improved by explicitly taking occlusion into account. Instead of trying to match a full CAD model to the point cloud, a pre-assignment filter similar to [49] could take only those CAD model points into account, which are expected to have a corresponding point in the sensor data and vice versa.

Furthermore, we need to improve the generation of the object hypotheses, especially in scenes where we detect planes that are larger than expected due to gap-free placement of several pieces of furniture. On top of this it could prove worthwhile to incorporate the color information provided by RGB-D cameras when merging planar regions. Last, we plan to automate the extraction of OWL-DL structural models for hypothesis generation from the CAD models.

Acknowledgments. This work is supported by the RACE project, grant agreement no. 287752, funded by the EC Seventh Framework Programme theme FP7-ICT-2011-7.

References

[1] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, W. Burgard, An evaluation of the RGB-D SLAM system, in: Proc. ICRA-2012, St. Paul, Minnesota, 2012.

[2] P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox, RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments, IJRR 31 (2012) 647–663.

[3] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, A. W. Fitzgibbon, KinectFusion: Real-time dense surface mapping and tracking, in: Proc. ISMAR, Basel, Switzerland, 2011, pp. 127–136.

[4] B. Kuipers, Y.-T. Byun, A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations, Robot. Auton. Syst 8 (1991) 47–63.

[5] A. Nüchter, J. Hertzberg, Towards semantic maps for mobile robots, Robot. Auton. Syst. 56 (2008) 915–926.

[6] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, P. Jensfelt, Search in the real world: Active visual object search based on spatial relations, in: Proc. ICRA-2011, Shanghai, China, 2011.

[7] L. Kunze, K. K. Doreswamy, N. Hawes, Using qualitative spatial relations for indirect object search, in: Proc. ICRA-2014, IEEE, 2014, pp. 163–168.

[8] H. Zender, Ó. M. Mozos, P. Jensfelt, G.-J. M. Kruijff, W. Burgard, Conceptual spatial representations for indoor mobile robots, Robot. Auton. Syst. 56 (2008) 493–502.

[9] C. Galindo, A. Saffiotti, Inferring robot goals from violations of semantic knowledge, Robot. Auton. Syst. 61 (2013) 1131–1143.

[10] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, 2nd ed., Cambridge University Press, 2007.

[11] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, OWL Web Ontology Language Reference, W3C Recommendation, W3C, 2004. URL: `http://www.w3.org/TR/owl-ref/`.

[12] M. Richardson, P. Domingos, Markov logic networks, Machine Learning 62 (2006) 107–136.

[13] D. Jain, Knowledge engineering with Markov Logic Networks: A review, in: Proc. 3rd Workshop on Dynamics of Knowledge and Belief (DKB 2011), 2011.

[14] D. Jain, S. Waldherr, M. Beetz, Bayesian Logic Networks, Technical Report, IAS Group, Fakultät für Informatik, Technische Universität München, 2009.

[15] M. Günther, T. Wiemann, S. Albrecht, J. Hertzberg, Model-based object recognition from 3D laser data, in: Proc. KI-2011, 2011, pp. 99–110.

[16] M. Günther, T. Wiemann, S. Albrecht, J. Hertzberg, Building semantic object maps from sparse and noisy 3D data, in: Proc. IROS-2013, Tokio, Japan, 2013, pp. 2228–2233.

[17] R. B. Rusu, Z. C. Marton, N. Blodow, M. E. Dolha, M. Beetz, Towards 3D point cloud based object maps for household environments, Robot. Auton. Syst. 56 (2008) 927–941.

[18] J. Mason, B. Marthi, An object-based semantic world model for long-term change detection and semantic querying, in: Proc. IROS-2012, Vilamoura, Portugal, 2012, pp. 3851–3858.

[19] D. Pangercic, B. Pitzer, M. Tenorth, M. Beetz, Semantic object maps for robotic housework - representation, acquisition and use, in: Proc. IROS-2012, Vilamoura, Portugal, 2012, pp. 4644–4651.

[20] B. Neumann, R. Möller, On scene interpretation with description logics, Image Vision Comput. 26 (2008) 82–101.

[21] J. Majumdar, A. G. Seethalakshmy, A CAD model based system for object recognition, J. Intell. Robotics Syst. 18 (1997) 351–365.

[22] C. Brenner, J. Böhm, J. Gühring, CAD-based object recognition for a sensor/actor measurement robot, IAPRS, HAKODATE 32 (1998) 209–216.

[23] F. Bosché, Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction, Adv. Eng. Inform. 24 (2010) 107–118.

[24] U. Klank, D. Pangercic, R. B. Rusu, M. Beetz, Real-time CAD model matching for mobile manipulation and grasping, in: 9th IEEE-RAS Intl. Conf. Humanoid Robots, Paris, 2009.

[25] O. M. Mozos, Z. C. Marton, M. Beetz, Furniture Models Learned from the WWW – Using Web Catalogs to Locate and Categorize Unknown Furniture Pieces in 3D Laser Scans, Robotics & Automation Magazine 18 (2011) 22–32.

[26] V. Usenko, F. Seidel, Z.-C. Marton, D. Pangercic, M. Beetz, Furniture classification using WWW CAD models, in: IROS'12 Workshop on Active Semantic Perception (ASP'12), Vilamoura, Portugal, 2012.

[27] W. Wohlkinger, A. Aldoma, R. B. Rusu, M. Vincze, 3DNet: Large-scale object class recognition from CAD models, in: Proc. ICRA-2012, St. Paul, Minnesota, 2012, pp. 5384–5391.

[28] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, A. J. Davison, SLAM++: Simultaneous localisation and mapping at the level of objects, in: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE, 2013, pp. 1352–1359.

[29] K. Lai, L. Bo, X. Ren, D. Fox, Detection-based object labeling in 3D scenes, in: Proc. ICRA-2012, 2012.

[30] J. J. Gibson, The ecological approach to visual perception, Houghton Mifflin, Boston, MA, USA, 1979.

[31] A. Chemero, M. T. Turvey, Gibsonian affordances for roboticists, Adaptive Behaviour 15 (2007) 473–480.

[32] T. Wiemann, K. Lingemann, A. Nüchter, J. Hertzberg, A toolkit for automatic generation of polygonal maps – Las Vegas Reconstruction, in: Proc. ROBOTIK-12, 2012, pp. 446–415.

[33] T. Wiemann, H. Annuth, K. Lingemann, J. Hertzberg, An evaluation of open source surface reconstruction software for robotic applications, in: Advanced Robotics (ICAR), 2013 16th International Conference on, IEEE, 2013.

[34] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the 4th Eurographics Symposium on Geometry Processing (SGP '06), Eurographics Association, 2006, pp. 61–70.

[35] N. Amenta, S. Choi, R. K. Kolluri, The power crust, in: Proceedings of the 6th ACM Symposium on Solid Modeling and Applications (SMA '01), ACM, New York, NY, USA, 2001, pp. 249–266.

[36] H. Edelsbrunner and E.P. Mücke, Three-Dimensional Alpha Shapes, ACM Transactions on Graphics 13 (1994) 43–72.

[37] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, Comp. Graph. 26 (1992) 71–78.

[38] J. Weingarten, R. Siegwart, 3D SLAM using planar segments, in: Proc. IROS-2006, IEEE, 2006, pp. 3062–3067.

[39] J. Biswas, M. Veloso, Planar polygon extraction and merging from depth images, in: Proc. IROS-2012, Vilamoura, Portugal, 2012.

[40] A. Trevor, J. Rogers, H. Christensen, Planar surface slam with 3d and 2d sensors, in: Proc. ICRA-2012, 2012, pp. 3041–3048.

[41] E. de Koning, Polyline Simplification Library (psimpl), 2010. `http://psimpl.sourceforge.net/`.

[42] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, World Wide Web Consortium, 2004. URL: `http://www.w3.org/Submission/SWRL`.

[43] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, J. Web Sem. 5 (2007) 51–53.

[44] P. Besl, N. McKay, A method for registration of 3–D shapes, IEEE T. Pattern Anal. Mach. Intell. 14 (1992) 239–256.

[45] S. Albrecht, T. Wiemann, M. Günther, J. Hertzberg, Matching CAD object models in semantic mapping, in: Proc. ICRA 2011 workshop: Semantic Perception, Mapping and Exploration, SPME '11, Shanghai, China, 2011.

[46] T. Wiemann, K. Lingemann, J. Hertzberg, Automatic map creation for environment modelling in robotic simulators, in: Proc. 27th European Conference on Modelling and Simulation (ECMS 2013), 2013, pp. 712–718.

[47] J. J. Kuffner, Effective sampling and distance metrics for 3D rigid body path planning, in: Proc. ICRA-2004, 2004, pp. 3993–3998.

[48] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, 6D SLAM – 3D mapping outdoor environments, J. Field Robot. 24 (2007) 699–722.

[49] S. May, R. Koch, R. Scherlipp, A. Nüchter, Robust registration of narrow-field-of-view range images, in: Proc. SYROCO '12, 2012.

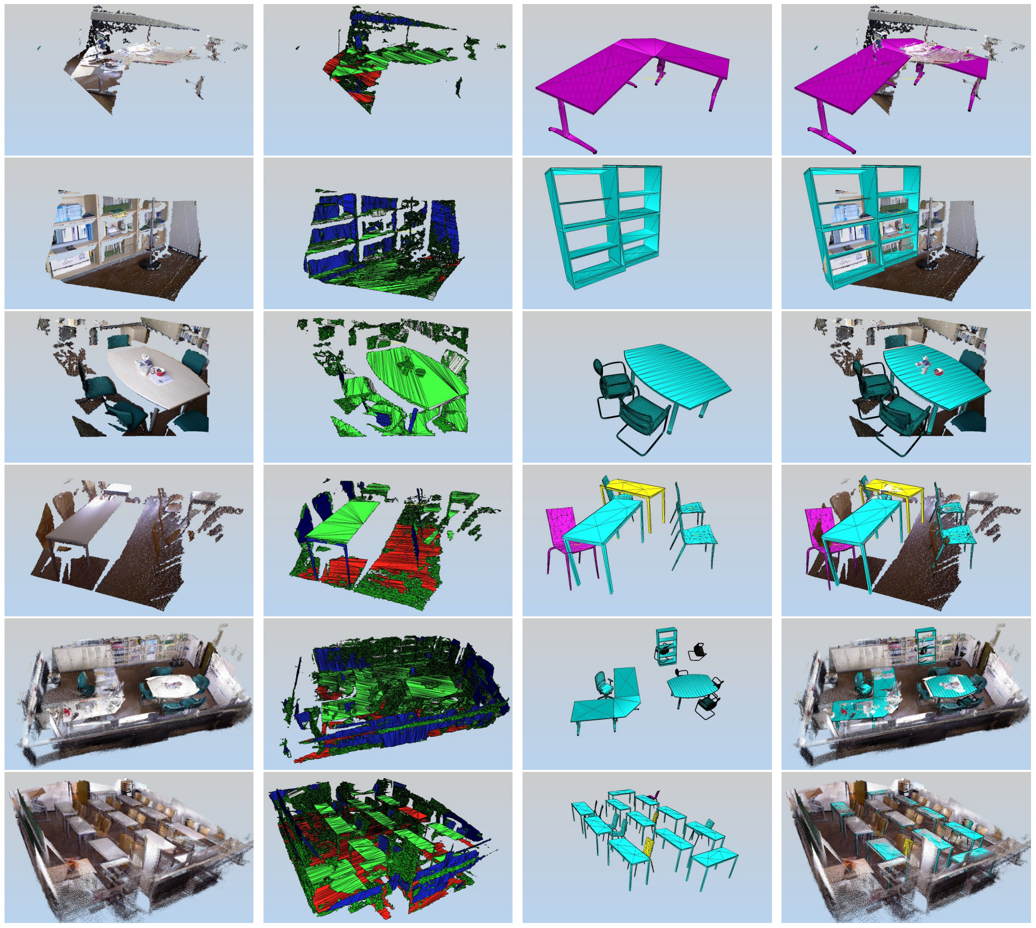

Figure 11: Results of each step in the processing pipeline. Columns, from left to right: (a) the original point cloud from a single Kinect frame; (b) the reconstructed triangle mesh; (c) sampled CAD models after ICP pose correction; (d) the point cloud overlaid with the recognized CAD models. Row 1: error in pose estimation due to only partial visibility of the desk; Row 2-3: correct placement of the recognized objects; Row 4: several good matches and one false positive. The color of the models indicates the quality of the recognition: cyan indicates a true positive, where the pose fits well with the actual pose, magenta is a true positive where the final pose is not well aligned and yellow shows a false positive. Rows 1-3 show data from single frames of the office dataset, row 4 from the seminar room dataset, while row 5 and 6 show the full scene of the office and seminar room datasets, respectively. (Figure based on [16], extended)

Appendix A. Vitae

Martin Günther is a PhD student in the Knowledge-Based Systems group at Osnabrück University. His research interests include AI-based robot control, semantic mapping and context information in perception.

Thomas Wiemann finished his PhD thesis on automatic generation of polygonal maps for robotic applications at the KBS group at Osnabrück University in 2013. An important aspect of his research is generating semantic scene interpretations from point cloud data.

Sven Albrecht is a PhD student in the KBS group at Osnabrück University. His main research interests are semantic mapping and 3D data interpretation.

Joachim Hertzberg is a full professor for computer science at Osnabrück University, heading the KBS group; he is head of the Osnabrück branch of DFKI's Robotics Innovation Center, too. His areas of interest are AI and Mobile Robotics, with a focus on plan-based robot control.