

Functional Object Mapping of Kitchen Environments

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Emanuel Dolha, Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{rusu,marton,blodow,dolha,beetz}@cs.tum.edu

Abstract—In this paper we investigate the acquisition of 3D functional object maps for indoor household environments, in particular kitchens, out of 3D point cloud data. By modeling the static objects in the world into hierarchical classes in the map, such as cupboards, tables, drawers, and kitchen appliances, we create a library of objects which a household robotic assistant can use while performing its tasks.

Our method takes a complete 3D point cloud model as input, and computes an object model for it. The objects have states (such as open and closed), and the resulted model is accurate enough to use it in physics-based simulations, where the doors can be opened based on their hinge position. The model is built through a series of geometrical reasoning steps, namely: planar segmentation, cuboid decomposition, fixture recognition and interpretation (e.g. handles and knobs), and object classification based on object state information.

I. INTRODUCTION

In this paper we investigate the automatic acquisition of functional environment models for household robotic assistants operating in indoor household environments, in particular kitchens. These models do not only allow the robot to localize itself and navigate, but are also resources that provide semantic knowledge about the static objects in the environment, what they are, and how they can be operated. Thus, the objects in the model are cupboards, tables, drawers, and kitchen appliances. The object models are structured and can have states. For example, a cupboard has a front door, handles and hinges, and



Fig. 1. Autonomous robot used for the experiments in this article. A SICK laser scanner is mounted on the left robot arm, which allows the robot to acquire 3D scans of the environment by sweeping its arm.

a storage place.

We set up our scenario as follows: the robot enters the room and repeatedly turns and makes a sweeping movement with its left arm that has a laser sensor mounted on its end effector (see Figure 1). Storing the individual laser readings during the sweep and combining them with the arm’s joint angles results in a 3D laser scan. The scan is converted to a point cloud where each point is represented by a tuple containing the 3D position in world coordinates.

After processing, integrating, and interpreting the individual scans, the robot transforms the Point Cloud Data (PCD) into a functional environment model as it is depicted in Figure 2. This map represent the objects that are relevant for the robot’s tasks – the cupboards, drawers, kitchen appliances, and tables, along with instructions on how to handle them.

For object classification of relevant furniture pieces, we assume that the pieces of furniture present in the environment are characterized by rectangular planar surfaces in the data. Furthermore, we assume that they have utilitarian functions, in the sense that they need to have additional fixtures such as handles and knobs to open and operate them. In a general sense, they can be viewed as containers with states (open or closed) where objects can be placed in or manipulated.

Under these assumptions, an appropriate computational structure for acquiring functional object maps of the environment includes the following steps:

- exploit the planarity of the environment structure in order to extract rectangular regions that constitute the frontal faces of the relevant environment objects;
- fit lines to the edges of the rectangular regions, and generate cuboid object hypotheses from these rectangles;
- classify the cuboid object hypotheses into cupboards and different type of kitchen appliances, based on the fixtures (handles and knobs) that can be segmented on their frontal planes;
- exploit the knowledge about object states, in particular that containers are open and closed to infer the correct object class.



Fig. 2. A functional map of the kitchen environment superimposed on the raw scan from which it was obtained.

We represent the obtained functional map using XML¹, such that they can be easily shared between different applications and robots, and that a robot can query the map using XQuery in order to retrieve necessary information from the model. The objects in our maps are instances of object classes where the object classes are organized in a specialization hierarchy. Thus, a cupboard is a specialization of a container and can therefore be used to store objects, and an oven is a specific kind of container that is used to prepare meals.

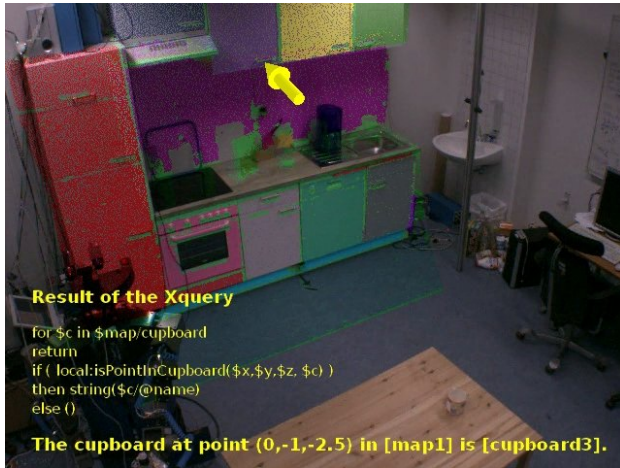


Fig. 3. Querying the object map using XQuery and the visualization of the results superimposed on a photograph of the scanned kitchen.

Environment models are resources for performing actions effectively and efficiently using the information of the model. For this reason, we can classify environment models by the types of queries they can answer. Most environment models, are made for answering the queries *where am I?*, *how do I get to my destination?*, and *do I collide if I perform this movement?*. On the other hand, the types of queries that functional object environment models are capable of answering also include: *what's the rightmost cupboard of the kitchenette?*, *where is the oven located?*, *which side does the cupboard door open to?*, or *which kitchen units store food?*. Figure 3 presents a query example for our functional map using XQuery.

The remainder of this paper is organized as follows. The next section (Section II) addresses related work relevant to our project. Section III briefly describes our system overview. Sections IV and V we present the geometrical reasoning modules that form the basis of our functional mapping approach. Section VI discusses the rule based system for building hierarchical object classes, followed by a discussion of the system's overall performance as well as one usage example in Section VII. We conclude and give insight on our future work in Section VIII.

¹The XML representation is only an intermediate representation for us. We intend to represent maps using OWL (Web Ontology Language). Using OWL and by importing knowledge from encyclopedic knowledge bases like OpenCyc, we can combine our environment object models with substantial amounts of knowledge about the objects contained in the environment model and what these objects can be used for.

II. RELATED WORK

Several efforts in the past have been made regarding the creation of environmental object maps out of 3D range data. Since the creation of such maps is a highly complex process and involves the combination of several algorithms, we will try to address the most relevant publications for our work below. Related work on particular dimensions will be addressed in their respective technical sections.

An EM-based algorithm for learning 3D models of indoor environments is presented in [1]. The maps are created using mobile robots equipped with laser range finders, but they do not include any semantic information. The work in [2] uses a stereoscopic camera system and a knowledge base in the form of a semantic net to form 3D models of outdoor environments. Two parallel representations, one spatial and one semantic, are proposed in [3] for an indoor environment, but their approach needs further investigation. An object based approach for cognitive maps is used to recognize objects and classify rooms in different categories in [4]. The work presented in [5] provides a method for classifying different places in the environment into semantic classes like doorways, kitchens, corridors, rooms using simple geometric features extracted from laser data and information extracted from camera data. The semantic interpretation in [6] takes into account generic architectural elements (floor, ceiling, walls, doors) which are identified based on the relationships between the features (parallel, orthogonal, above, under, equal height). Finally, a decomposition of maps into regions and gateways is presented in [7].

With few exceptions, in particular in the area of cognitive mapping [4], [8], but also including [9], [10], maps do not represent objects relevant for other robot tasks, besides navigation.

III. SYSTEM OVERVIEW

To obtain functional maps from individual raw scans, a complete and compact representation of the world has to be constructed first, that is the separate views have to be registered together and transformed into a global coordinate framework. We apply the methods presented in [11] for estimating point correspondences and registering the scans into a common model (see Figure 5). The resulted *complete* PCD model is considered the input of our functional mapping system. Figure 4 presents the overall system architecture.

Since the topic of this paper focuses on functional mapping, we will not address the modules outside its scope (namely Outlier Removal, Feature Estimation, Registration, and Resampling), as they have already been covered in our previous work [11]. After a complete PCD model has been obtained, it will go through the following processing steps:

- a region segmentation step for extracting planar regions, based on estimated surface normals and curvatures (Section IV);
- a model fitting step for identifying object candidates (cupboards, tables, etc.), represent them using cuboids, and search for connected handles and knobs (Section V);

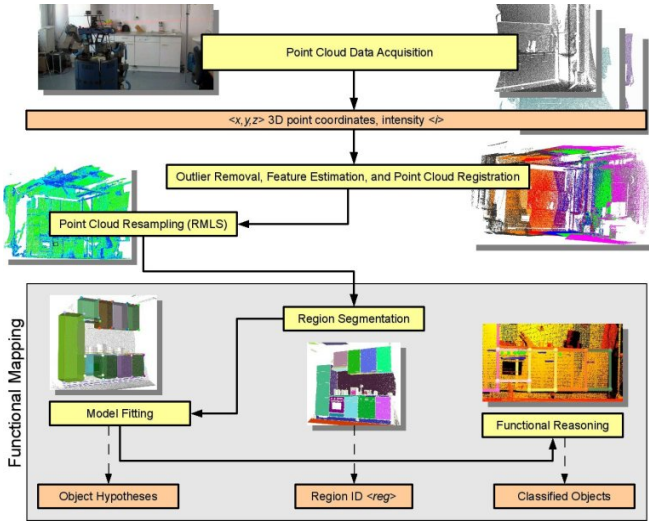


Fig. 4. Brief overview of the overall system architecture.

- a functional reasoning step where object candidates are verified and their functionality is deduced from the geometry data (Section VI).



Fig. 5. Two separate scans of the kitchen (left), and the resulted model after registration (right).

The region segmentation module encapsulates a region growing algorithm which propagates on directions of low curvature. Therefore, we require that surface curvature changes at each point are correctly estimated. After registration, the point cloud density will vary in the complete model from overlapping areas to non-overlapping ones. Additionally due to small registration errors, some regions might appear as double, which leads to incorrect curvature estimates. Changing the radius for the k -neighborhood selection (see [11] for details) will not help as edges will be smoothed out. This justifies the usage of the Point Cloud Resampling (RMLS) module right after registration (see Figure 6).

IV. REGION SEGMENTATION

Given an unorganized complete point cloud model P , containing $\{x, y, z\}$ 3D point coordinates, and normal and curvature estimates $\{n_x, n_y, n_z, c\}$, the problem is to find

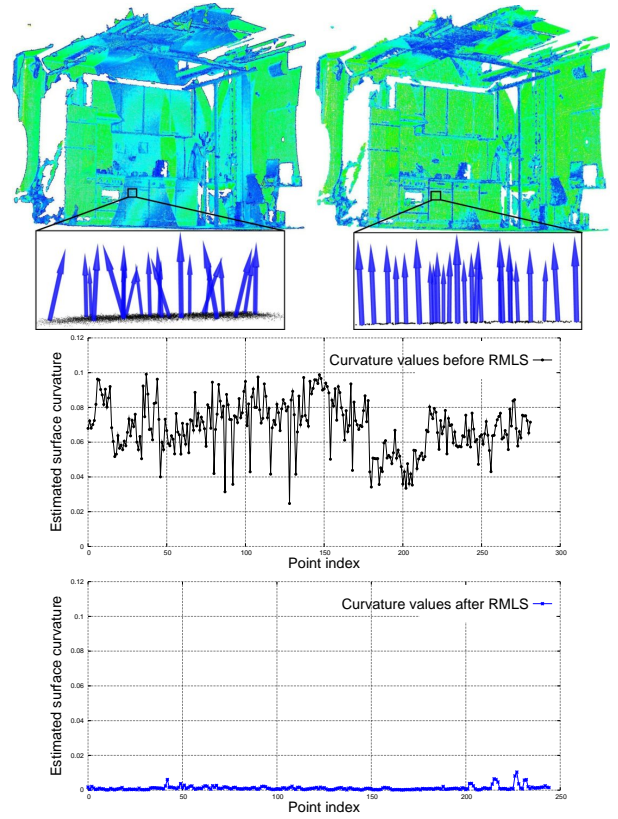


Fig. 6. Resulting surface curvature estimates (in logarithmic scale) on: a PCD after registering two partial views (top left); the resampled PCD (top right). Notice the errors in the surface normal estimates. The plots below depict the estimated surface curvatures before and after resampling.

connected regions with smooth surfaces that are separated by edges, i.e. sudden changes in curvature and normal orientation. Since the regions that we are interested in are in general planar surface patches, the segmentation module extracts these patches from the point cloud for further geometrical analysis.

To segment a given point cloud into areas of interest, we use a region growing approach based on smoothness constraint similar to [12]. The algorithm initially picks a point p with the lowest curvature (we call this the seed point) and looks at its k closest 3D neighbors. Each neighbor is individually verified whether it belongs to the same region as the seed point and whether it should be considered as a future seed point itself at a future iteration of the algorithm.

The criteria based on which a neighbor p_k is said to belong to the same region is:

$$\text{acos}(\langle n_p, n_{p_k} \rangle) \leq \alpha_{th}, \quad (1)$$

where n_p and n_{p_k} are the normalized normals of the seed point p and the neighbor p_k . Therefore, if the angle between their normals does not exceed a given threshold α_{th} , the point p_k is added to the current region and deleted from the point cloud. Then, if its estimated curvature is bound by a predefined c_{th} curvature, p_k will be pushed to the queue of seed points, so that its neighbors will be explored too, and

the procedure is restarted.

This method is similar to a region growing approach, with the difference that we propagate along directions of lower curvature first. The algorithm stops adding points to the current region if none of the neighbors fulfills the angle criteria and the queue of seed points is empty (meaning that all points with a small curvature were already considered).

We have extended the algorithm by automatically computing an optimal value of c_{th} , which depends on the curvatures found in the remaining point cloud. If only very flat regions are contained in the point cloud, a small curvature could already represent an edge, whereas in a point cloud with a high variation in curvature, c_{th} should be chosen bigger to avoid over-segmentation into many small patches. Therefore, at each step in our algorithm we sort the points by curvature and use the curvature at $x\%$ of that list. That means $(100 - x)\%$ of the points will have a curvature that will be considered *edge*. In this step, we also determine the seed point, as it is the first point in this list.

It should be noted that c_{th} (and thus x) does not influence if a point is merged with a region or not, it just filters which of the merged points will be considered as future seed points. This means that the value of x should be selected as a value lower than the approximate percentage of points that are expected to be on planar areas, but it will give correct results for a wide range of values, as c_{th} is recomputed (raised) in each step after the points with the lowest curvature are removed.

Because the resampled point cloud used as input data had smooth changes between surface normal directions, various levels of α_{th} did not influence the results significantly. Therefore, we fixed $\alpha_{th} = 5^\circ$, and only adapted c_{th} automatically based on the point cloud data. Experimentally, we determined that any cutoff value higher than $x = 50\%$ works fine, mostly because of the high number of planar areas in our datasets.

This assumption of the planar regions' dominance is verified in Figure 7. The curvatures in a scan of the kitchen (top) are presented in a histogram (middle), showing that most of the points have a very low curvature. Surface curvatures at point p are computed using the formula:

$$c = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \quad \lambda_0 < \lambda_1 < \lambda_2 \quad (2)$$

i.e. by comparing the 3 eigenvalues λ_i obtained from Principle Component Analysis of the local neighborhood of the query point p [11]. These values encode the elongation along the piecewise perpendicular axes of smallest/medium/biggest variations in point coordinates, and thus:

$$c = 0 \Leftrightarrow \lambda_0 = 0$$

Because $\lambda_0 = 0$ is possible only if all points in the neighborhood lie on the same plane, even a single outlier in the vicinity of p results in $c \geq 0$ for that point. Still, a value below ≈ 0.01 describes a point that is on a plane when inspected visually.

Experimentally, we determined the cutoff value of x to be 75%, as we expected many planar areas in a kitchen,

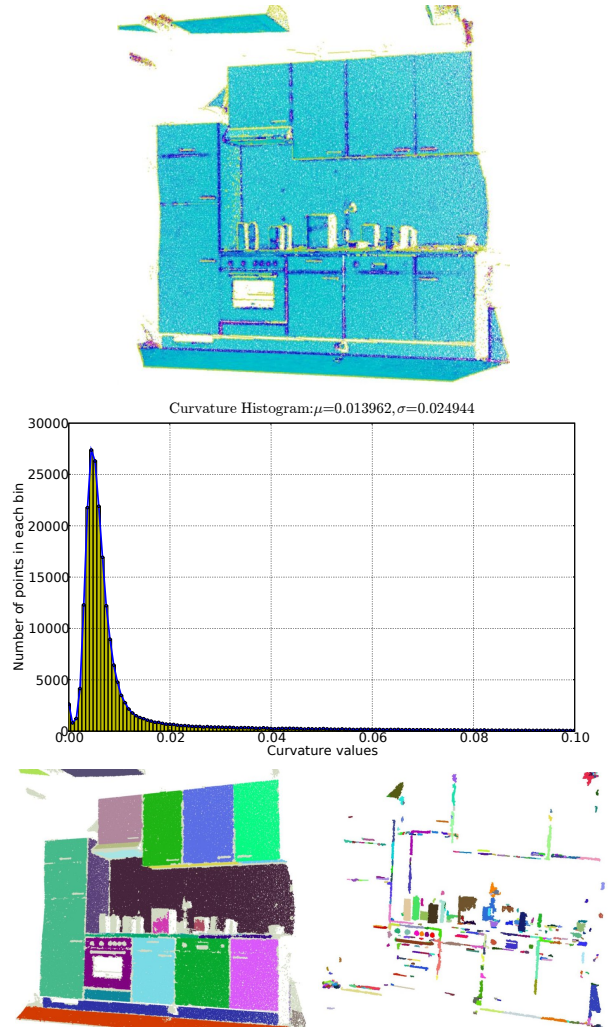


Fig. 7. Estimated surface curvatures for the kitchen scene (top); Curvature histogram plot (middle); Segmentation results for the kitchen: accepted regions (bottom left) and remaining points (bottom right).

and indeed, we found approximately 85% of the points to be located on planar regions. A value above this percentage would have produced under-segmentation as many regions would have been merged, and using a value around 50% would have resulted in oversegmenting the regions. This large interval for correctly setting x is possible due to the clear curvature difference between points on a planar region and the rest of the points.

Figures 7 (bottom left) and 8 present segmentation results for two different kitchen datasets.

After projecting the inliers on the model plane, we triangulate them and compute the total area of the region by summing the areas of the triangles. The regions which have a small area will be trimmed and only prospective regions will be further considered.

V. MODEL FITTING

Once a set of planar regions have been identified in the input data, we locate object candidates (Subsection V-A)

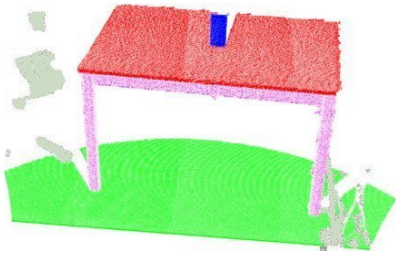


Fig. 8. Segmenting tables with $\alpha_{th} = 5^\circ$ and $x = 75\%$.

and the corresponding handles and knobs, if there are any (Subsection V-B).

A. Cuboid Fitting

From the set of triangulated polygons that replace 2D planar surfaces, the cuboid fitting module has to select those which are vertical and horizontal and satisfy a size criteria, and approximate them to 3D rectangular parallelepipeds (cuboids), by projecting the vertical polygons onto the walls (e.g. cupboards) and the horizontal ones on the floor (e.g. tables) of the room.

One of the most commonly used methods for computing the bounding 2D polygonal surface of a planar patch of points is to compute its convex hull. Since the majority of our planar surfaces are convex (e.g. frontal cupboards faces), computing convex hulls yields very fast and stable solutions. However, since we would like to generalize our methods in the future to other types of planar surfaces, such as tables, which are not always convex in shape, we decided to use an alternative, more robust approach. We start by analyzing the 2D planar patch, and identify its boundary points [11]. The results are presented in the left side of Figure 9. Once the list of all points on the boundary are obtained, they can easily be connected together to form the polygonal approximation of the patch.

For particular solutions, such as replacing regions of points with 2D quads, we use a MLESAC (Maximum Likelihood Estimation SAmple Consensus) [13] technique to robustly fit the best 4 lines to the region boundary and discard the outliers. The problem is reduced to minimizing the negative log likelihood $-L$ of all hypotheses:

$$-L = - \sum_{i=1}^N \log \left(\gamma \left(\frac{1}{\sigma \sqrt{2\pi}} \cdot e^{-\frac{r_i^2}{2\sigma^2}} \right) + \frac{1}{w} (1 - \gamma) \right) \quad (3)$$

with the probability of finding a good model being:

$$\Pr(M) = 1 - (1 - \gamma^p)^m, \quad (4)$$

where w represents the bounding box diagonal of the quad, and r_i the distance of point p_i to the model.

We intersect the pairs of lines which are close to perpendicular and we accept the region as a cupboard door or tabletop if we get 4 intersection points that form a 2D quad with reasonable dimensions. We force the edges of these 2D quads to align with the global coordinate system axes, which

are perpendicular on the floor and walls. Since the input point cloud represents data up to an entire room, the walls can be approximated with the faces of the bounding box. The quads which satisfy a required distance criteria are projected onto the floor and walls of the room to form cuboids. The results are presented in Figure 9 (bottom right) and constitute the basis for our functional mapping algorithm.



Fig. 9. Boundary points of each segmented planar region (top); 4 best fitted lines to each region boundary (bottom left); Cuboids created by projecting the 2D quad on the wall (bottom right).

The method works for vertical planar regions (cupboards, drawers), identifying candidates which have to be investigated further (e.g. locating handles and knobs and estimating opening trajectories). However, in cases where the surfaces (e.g. tables) are cluttered with additional objects, there isn't a big enough consistent region so our algorithm can fail, but scanning a table with only a few occluding objects on it will produce a correct segmentation (see Figure 8).

B. Extracting Handles and Knobs

The handles and knobs extraction module verifies in the list of segmented cuboids whether geometric shapes such as handles or knobs can be found in the proximity of their surfaces.

For finding out the location of handles and knobs, we look at clusters of points which did not get assigned to one of the segmented planar regions (see Figure 10), but all whose neighbors are inside a planar region. We select them for analysis if they are at a distance $dist_i$ from the surface, in between some predefined threshold

$$d_{min} \leq dist_i \leq d_{max},$$

where one would expect a handle. After projecting the cluster onto the plane, and computing the boundary points of each cluster, the search for a candidate shape will begin. Our

robust shape fitting method is based on a MLESAC estimator for finding lines and circles, followed by a non-linear LM (Levenberg-Marquardt) optimization of the shape's parameters.

As it can be seen in Figure 10, the scanning quality of the PCD is high enough to locate and correctly identify handles and knobs, even though some regions contain more handles due to the under-segmentation of those areas. These inconsistencies could be corrected by a better segmentation based on higher resolution scans, triggered by observing multiple shape candidates (handles) in a region. For example, detecting two handles for one cupboard could mean that there might be in fact two cupboards present, but under-segmentation occurred.

Higher resolution scans of specific areas would also be useful for 3D shape classification, either by fitting shape primitives like cylinders instead of lines, or by identification based on predefined CAD models[14], [11].

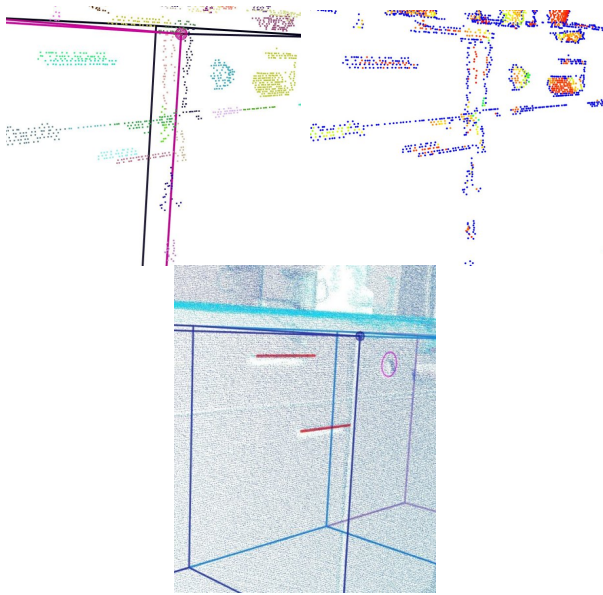


Fig. 10. The remaining clusters of points projected on their associated cuboids (top left); Estimated boundary points of each cluster (top right); Handles and knobs segmentation via line and circle fitting (bottom).

Please note that the clusters in the proximity of the detected handles are located just above the plane defining the region. They are the result of the sensors' inherent measurement errors, by averaging multiple return pulses. However, they are easy to filter using our method, since they do not satisfy the distance criteria and are not connected to the handles.

VI. FUNCTIONAL REASONING

Having a set of geometric primitives, more exactly horizontal planes and cuboids with connected handles and knobs, the next step is to verify whether they belong to a certain object class or not. The resulted hierarchical object model (see Figure 12) is constructed by testing the following con-

ditions on each $\$cuboid$ in the *for* $\$cuboid$ in $\$map/candidate$ XQuery loop:

- $hasKnobs(\$cuboid)$ – true if a cuboid has at least one knob associated with it;
- $hasHandle(\$cuboid)$ – true if a cuboid has at least one handle associated with it;
- $countKnobs(\$cuboid)$ – the number of knobs associated with the cuboid;
- $countHandles(\$cuboid)$ – the number of handles associated with the cuboid;
- $handleLocation(\$cuboid)$ – can only be applied for cuboids containing one handle and returns one of the following labels for its location: 'center' – if the handle is located roughly in the middle of the planar surface's geometrical center; 'near-edge (position)' – the hinge of the door is located on the opposite edge.

The initial implementation of our hierarchical classification system is presented below. Note that while for the moment it uses hardcoded heuristics for object classes, our most current implementation investigates the possibility of learning the rules for classification using decision trees from multiple training examples.

- Cuboid: *Container*
 - with knobs (i.e. having at least one knob): *Kitchen Appliance*
 - * single knob and optionally one handle: *Dish Washer*
 - * multiple knobs and optionally one handle: *Oven*
 - * multiple handles: *under-segmented*
 - without any knobs: *Storage Space*
 - * single handle in center: *Drawer*
 - * single handle near an edge: *Cupboard* (with opening hinge being the opposite edge)
 - * multiple handles: *under-segmented*
 - * without any handles: *unidentified container*
- Horizontal planar region: *Table* (if verifies height and size criteria)

Figure 11 presents four identified object classes in the kitchen. Using the above criteria, from left to right: (i) the cuboid with multiple knobs is classified as the oven; (ii) an object with multiple handles and no knobs is possibly under-segmented and requires further analysis; (iii) the cuboid with one knob is classified as the dishwasher; and (iv) a container with one handle is determined to be a cupboard, which opens to the right, because of the handle's placement on the upper left side.

In those cases where the object candidate is ambiguous (under-segmented – see Figure 2), we infer that not enough data was available for segmentation. This should trigger the robot to perform additional higher resolution scans in those areas for further processing.

Saving the acquired information in XML format makes it easier for us to query the map for answers about the location, number of objects and the relationships between them (see Figure 3).

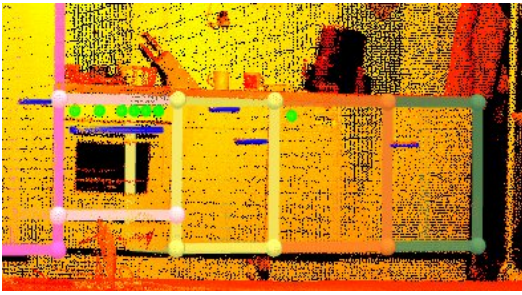


Fig. 11. Handles and knobs segmentation for object hypotheses in the kitchen.

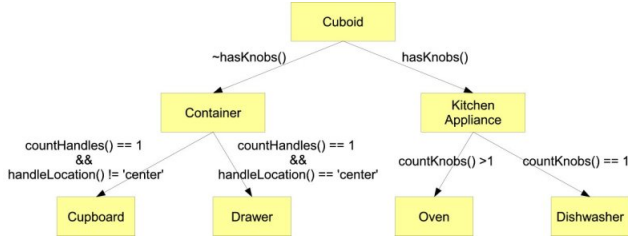


Fig. 12. Hierarchical object model.

The resulting model can be verified by testing it against ground truth, e.g. if a cupboard is present at the predicted location, try to open it and observe the results. In order for the robot to be able to open a cupboard or a drawer, an opening trajectory needs to be approximated, based on the position of the handle.

Another approach for testing the validity of the classified objects in the environment model, is to notice temporal differences in a region by scanning it at different time intervals. Figure 13 shows a snapshot of the registration process for two point clouds taken at different instances in time, together with the registration results. Here the robot observed the changes in the region of a cuboid, detecting the difference produced by a cupboard door being opened. This verifies the assumptions about the cupboard’s location and opening direction. These subsequent views are faster to obtain as the scan resolution can be decreased and only smaller regions of interest need to be captured.

VII. DISCUSSIONS AND FUNCTIONAL MAP USAGE

With regards to time constraints, we view the acquisition of a functional map as part of the deployment of a robot in a new environment, thus it only has to be done once before the robot starts performing its job. Once an initial model is built, changes in the map are easier to incorporate and require less time to compute. Therefore, while we are striving to improve our algorithms constantly both in terms of accuracy and robustness but also in execution time, we consider that for our application it’s not necessary to achieve real-time performance. While most of the modules in our processing pipeline are already within reach of realtime execution, their speed performance usually depends on the desired output accuracy. In particular, faster desired execution time means

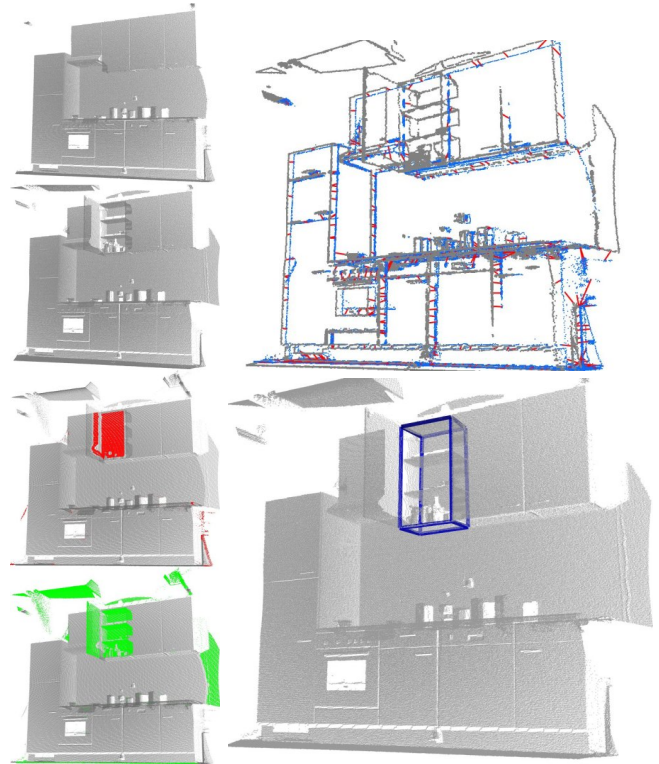


Fig. 13. From top to bottom and left to right: two scans of the kitchen scene depicting two states of a cupboard (closed and open), a snapshot from the registration process, and the highlighted temporal differences and segmented container.

that we would have to work with more coarse grained resolutions, which in several cases would not allow us to get the fine details we need.

We have applied the system for acquiring an environment model in a complete kitchen laboratory. The system built accurate models of the environment. The parts of the environment that couldn’t be modeled were mostly areas with shiny surfaces such as the oven door, the kitchen sink, and the windows. This is due to the inherent limitations of the laser sensing devices.

The objects satisfying our object specifications were recognized correctly, besides those that were largely occluded by other objects. To account for these problems, we intend to investigate the use of complementary sensors including cameras, as well as active map acquisition. We further intend to improve our recognition algorithms to better deal with large occlusions.

In the final model a refrigerator will be classified as a large cupboard as it is hard to make any difference between them, especially because latest kitchen design models usually place it inside a cupboard (like in our experimental setup).

In these cases the robot could infer the location of the fridge either by assuming it to be the largest cupboard or observing its usage. This however is not a major drawback as a fridge is indeed a cupboard-like container, with the difference that only specific types of items are stored inside.

The final model in XML format can be imported into a 3D

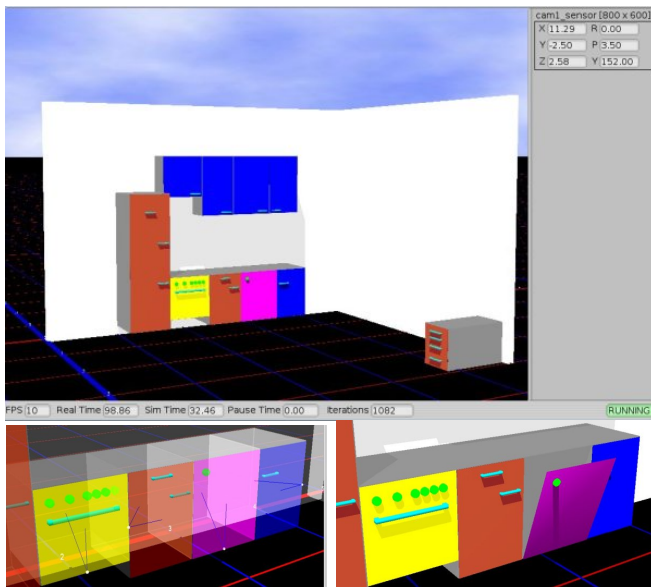


Fig. 14. Functional object map in the Gazebo 3D simulator.

simulation environment (Gazebo) as seen in Figure 14, where models of kitchens were already used for task planning and optimization [15].

VIII. CONCLUSIONS AND FUTURE WORK

We have presented a system for building functional maps of indoor household environments (i.e. kitchens). Relevant kitchen objects including cupboards, kitchen appliances, and tables were correctly recognized and localized, the cuboid-like objects were classified into different classes based on a set of high-level features: *hasHandle()*, *hasKnobs()*, *countHandles()*, *countKnobs()*, and *handleLocation()*, giving promising results for applications such as ours.

We are currently investigating ways of adding more semantic information to our maps, by making use of sensor networks as we presented in [15], [16], and plan to further extend our current results by using techniques such as active mapping (i.e. automatically detect when an area needs to be rescanned or scanned at a higher resolution).

For example, using an RFID tag on the hand of the robot and deploying RFID readers inside the cupboards, and magnetic sensors on cupboard doors (signaling openings and closings of the cupboard), can be combined with 3D information coming from the laser in order to draw conclusions about the connection between them (e.g. RFID antenna A is located in cupboard at coordinates XYZ , whose opening/closing is signaled by magnetic sensor M)[16].

Finally, our long term goal is to recognize other objects such as kitchen utensils based on a larger vocabulary of geometric primitives. We've already made the first steps in [11], but further integration and testing still needs to be done.

An important step we are currently working on, is the replacement of the empirically fixed rule-based classification with a probabilistic one, based on previous observations. These observations can be obtained by supervised learning

techniques in a simulation environment, where we have access to a wide range of real kitchens models (courtesy of Plantek and Segmüller department stores). These models can be scanned in simulation and given as input to our algorithm, and the results used as training data for a decision tree classifier.

IX. ACKNOWLEDGMENTS

This work is supported by the CoTeSys (Cognition for Technical Systems) cluster of excellence.

REFERENCES

- [1] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun, "Using EM to Learn 3D Models of Indoor Environments with Mobile Robots," in *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 329–336.
- [2] O. Grau, "A scene analysis system for the generation of 3-D models," in *NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, 1997, p. 221.
- [3] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, and et al., "Multi-Hierarchical Semantic Maps for Mobile Robotics."
- [4] S. Vasudevan, S. Gächter, V. Nguyen, and R. Siegwart, "Cognitive maps for mobile robots-an object based approach," *Robotics and Autonomous Systems*, vol. 55, no. 5, pp. 359–371, 2007.
- [5] O. M. Mozos, A. Rottmann, R. Triebel, P. Jensfelt, and W. Burgard, "Semantic Labeling of Places using Information Extracted from Laser and Vision Sensor Data," in *In Proceedings of the IEEE/RSJ IROS Workshop: From sensors to human spatial concepts*, Beijing, China, 2006.
- [6] A. Nuechter, H. Surmann, and J. Hertzberg, "Automatic Model Refinement for 3D Reconstruction with Mobile Robots," 2003.
- [7] D. Schröter, T. Weber, M. Beetz, and B. Radig, "Detection and Classification of Gateways for the Acquisition of Structured Robot Maps," in *Proc. of 26th Pattern Recognition Symposium (DAGM), Tübingen/Germany*, 2004.
- [8] J. Modayil and B. Kuipers, "Bootstrap learning for object discovery," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-04)*, 2004, pp. 742–747.
- [9] D. Schröter and M. Beetz, "Acquiring Modells of Rectangular Objects for Robot Maps," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA), New Orleans/USA*, 2004.
- [10] R. Triebel, Óscar Martínez Mozos, and W. Burgard, "Relational Learning in Mobile Robotics: An Application to Semantic Labeling of Objects in 2D and 3D Environment Maps," in *Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications (GfKI)*, Freiburg, Germany, 2007.
- [11] R. B. Rusu, N. Blodow, Z. Marton, A. Soos, and M. Beetz, "Towards 3D Object Maps for Autonomous Household Robots," in *Proceedings of the 20th IEEE International Conference on Intelligent Robots and Systems (IROS), San Diego, CA, USA, Oct 29 - 2 Nov., 2007*.
- [12] T. Rabbani, F. van den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," in *IEVM06*, 2006.
- [13] P. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.
- [14] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, June 2007.
- [15] M. Beetz, J. Bandouch, A. Kirsch, A. Maldonado, A. Müller, and R. B. Rusu, "The assistive kitchen — a demonstration scenario for cognitive technical systems," in *Proceedings of the 4th COE Workshop on Human Adaptive Mechatronics (HAM)*, 2007.
- [16] R. B. Rusu, B. Gerkey, and M. Beetz, "Robots in the kitchen: Exploiting ubiquitous sensing and actuation," in *Robotics and Autonomous Systems Journal (Special Issue on Network Robot Systems)*, accepted for publication, 2007.