Towards 3D Object Maps for Autonomous Household Robots

Conference Paper · October 2007

DOI: 10.1109/IROS.2007.4399309 · Source: DBLP

5 authors, including:

Radu Bogdan Rusu
Open Perception
67 PUBLICATIONS **12,256** CITATIONS
SEE PROFILE

Nico Blodow
Technische Universität München
31 PUBLICATIONS **5,323** CITATIONS
SEE PROFILE

Zoltan Csaba Marton
German Aerospace Center (DLR)
101 PUBLICATIONS **4,184** CITATIONS
SEE PROFILE

Michael Beetz
Universität Bremen
486 PUBLICATIONS **14,475** CITATIONS
SEE PROFILE

Towards 3D Object Maps for Autonomous Household Robots

Radu Bogdan Rusu, Nico Blodow, Zoltan Marton, Alina Soos, Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{rusu, blodow, marton, soos, beetz}@cs.tum.edu

***Abstract*— This paper describes a mapping system that acquires 3D object models of man-made indoor environments such as kitchens. The system segments and geometrically reconstructs cabinets with doors, tables, drawers, and shelves, objects that are important for robots retrieving and manipulating objects in these environments. The system also acquires models of objects of daily use such glasses, plates, and ingredients. The models enable the recognition of the objects in cluttered scenes and the classification of newly encountered objects.**

Key technical contributions include (1) a robust, accurate, and efficient algorithm for constructing complete object models from 3D point clouds constituting partial object views, (2) feature-based recognition procedures for cabinets, tables, and other task-relevant furniture objects, and (3) automatic inference of object instance and class signatures for objects of daily use that enable robots to reliably recognize the objects in cluttered and real task contexts. We present results from the sensor-based mapping of a real kitchen.

I. Introduction

Maps (or environment models) are resources that enable robots to better perform their tasks. Using maps robots can better plan their own activities and recognize, interpret, and support the activities of other agents in their environments [1]. Most robot maps acquired and used so far primarily enable robot localization and navigation [2]. With few exceptions, in particular in the area of cognitive mapping [2], [3], but also including [4], [5], maps do not represent objects relevant for other robot tasks. Objects are in most cases geometric primitives such as lines and planes that make maps more compact and abstract [6].

In contrast, robots that are to perform manipulation tasks in human living and working environments need much more comprehensive and informative object models. Consider, for example a household robot that is to set tables. Such a robot must know that cups and plates are stored in cabinets, that cabinets can be opened to retrieve the objects inside, that the visual system cannot see the objects inside a cabinet unless the door is open. It must also know in *which* cabinets given objects are stored to not have to search for them. In addition, the map should also enable the robot to easily recognize the objects of interest in its work space.

Our research agenda aims at investigating the representations, the acquisition, and the use of robot maps that provide robots with these kind of knowledge about objects and thereby enable autonomous service robots, such as household robots, to perform their activities more reliably and efficiently.

This paper presents substantial steps towards the realization of such maps and the respective mapping mechanisms. Conceptually, we perform a feasibility study that demonstrates how robot maps containing (1) object representations for objects such as cabinets, tables, drawers, etc and (2) object libraries for objects of daily use can be represented and acquired. The main technical contributions of this paper are the following ones: (1) a novel robust, accurate, and efficient algorithm for constructing complete object models from 3D point clouds constituting partial object views, (2) feature-based recognition procedures for cabinets, tables, and other task-relevant furniture objects, and (3) automatic inference of object instance and class signatures for objects of daily use that enable robots to reliably recognize the objects in cluttered and real task contexts.

The remainder of the paper is organized as follows. The next section introduces our map representation and gives an overview of the mapping process. Section III describes the theoretical algorithms as well as their implementation. Sections IV and V describe the computational mechanisms for acquiring the environment maps and the object libraries respectively, as well as the experimental results obtained. We conclude with a short discussion of related work, our conclusions, and a sketch of our future work.

II. Environment Maps and their Acquisition

In this section we first conceptualize a kitchen from the view of a household robot that is to perform tasks such as setting the table, preparing meals, and cleaning up. We then describe how we represent this conceptualization as a robot map. We finally consider variants of computational problems for acquiring such robot maps and give an overview of the mapping program we implemented.

Fig. 1. Point clouds representing objects.

A. Kitchens through the Eyes of Household Robots

Looking through the eyes of household robot the kitchen is a room that essentially consists of a set of cabinets, containers with front doors, shelves, which are open containers, and tables — horizontal planes that can be used for performing

kitchen work, having meals, etc. Some of the cabinets have specific roles: the fridge keeps food cold, the oven and microwave is used to cook meals, the dishwasher to clean the dishes after use. There might be more pieces of furniture that have important roles for the robots' tasks, but for now we only consider these ones. All other types of furniture, such as chairs are uniformly considered to be obstacles.

The kitchen also contains smaller objects with changing locations, such as food, ingredients, cooking utensils, and small appliances such as coffee machines. For these objects the robot needs libraries of object representations that can be used to find and recognize these objects, to keep track of their status, etc (see Figure 1).

B. Object Maps of Indoor Environments

Our map representation is an extension of RG (Region-Gateway) maps proposed by Schröter et al. [4]. RG maps are tuples $\langle R; G \rangle$, where *R* denotes a set of regions and *G* is a set of gateways that represent the possible transitions between regions. A region has a class label ("office-like" or "hallway-like"), a compact geometric description, a bounding box, one or two main axes, a list of adjacent gateways, and the associated objects *O*.

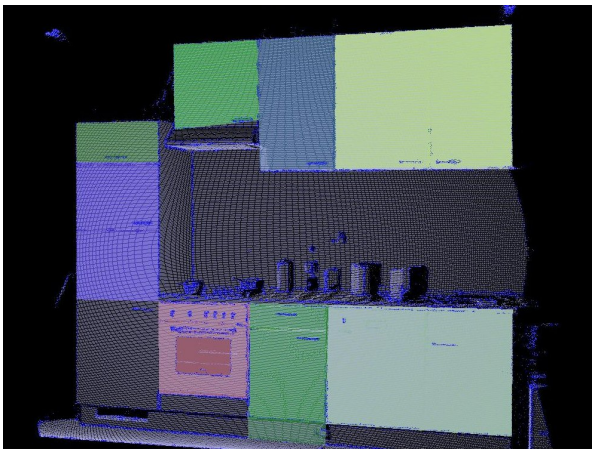

Fig. 2. Depiction of segmented cupboards in a kitchen.

In this paper we consider how the objects *O* are represented and how the models of these objects are acquired. The objects *O* are a tuple $\langle F; OL \rangle$, where *F* is the set of "furniture-like" objects and *OL* is the object library. The "furniture-like" objects *F* is a set of objects that include geometric 3D models, position and orientation, and an object class label. Object classes are cabinets, drawers, shelves, tables, and obstacles, where obstacles are all the objects that do not fall into the first four classes. Cabinets are cuboid containers with a front door that might contain objects of the object library *OL*. Some cabinets have specific purposes such as the oven, the refrigerator, and the dishwasher. The other cabinets are used to store objects. Tables are another important subclass of furniture-like objects. They are horizontal rectangular flat surfaces approximately at hip height that satisfy some given minimal size and length requirements. Similarly, we have drawers and shelves as additional furniture-like object classes.

The object library contains the small appliances and the objects of daily use, such as ingredients, cups, plates, pots, etc. These objects are moved around and might change their states: cups can be full, empty, used, to be used, clean, etc. These objects are indexed spatially, functionally and as affordances.

C. The Mapping Problem

In general the mapping problem is to infer the semantic object map that best explains the data acquired during the mapping process. The sensor data is a sequence $\langle v_i, \text{pose}_i \rangle$ where v_i is a point cloud where each point has an $\langle x, y, z \rangle$ coordinate and possibly additional perceptual features associated with it, such as its color. The scans cover the environment to be mapped and each point cloud has sufficiently large overlap with previous scans such that it can be roughly spatially aligned with previous scans. pose_i specifies the position and orientation of the recording sensor and might be unknown if the robot cannot estimate the position of its sensor.

The output consists of a compact obstacle representation of the environment. In addition, the system is to represent the cabinets and drawers that it sensed in their closed as well as opened state as objects with the respective furniture category. The system also has to represent the tables and shelves explicitly and label them respectively.

given: $\{ \langle v_1, \text{pose}_1 \rangle, ..., \langle v_n, \text{pose}_n \rangle \}$
estimate: a compact and accurate polygonal representation of $v_1 \cup ... \cup v_n$ biased towards planar rectangular surfaces
infer: object models for cabinets, drawers, shelves, and tables.

D. Overview of the Mapping System

The operation of the mapping system described in this paper is depicted in Figure 3.

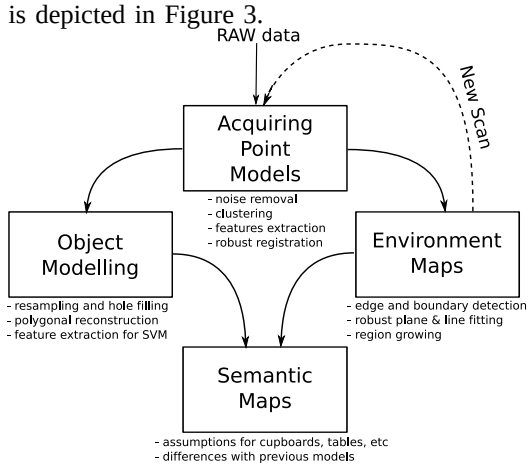

Fig. 3. The architecture of our system.

The first module *Acquiring Point Models* constructs comprehensive and accurate point models of the individual views provided by the sensor. The point models are then further processed by the type specific mapping components: the *Object Modelling* and the *Environment Mapping* module. The object and furniture models returned by these models are

then combined into the *Semantic Map*. The *Acquiring Point Models* module first classifies individual point measurements with respect to whether or not they are outliers. Outliers are removed. After outlier removal measurement points are clustered and additional features are extracted to make the consistent integration of different views more efficient and more robust. The integration is done by an iterative optimization algorithm that tries to minimize the distance of points that are believed to correspond to the same 3D points. The details of the algorithm are explained in Section III.

The steps *Object* and *Environment Modelling* are specific to the aspect of the environment that is mapped. *Object Modelling* acquires models of the objects of daily use (see Figure 1). *Object modelling* smoothes object surfaces and reconstructs occluded surfaces through point resampling. The resulting point cloud is polygonally reconstructed and object classifying and identifying features are inferred. *Environment Modelling* estimates polygonal representations but also recognizes and analyzes large rectangular planes. Special purpose "sensing" routines look for voxel clouds occluded by rectangular planes — the cabinets, for large horizontal rectangular planes — the tables, shelves, and drawers.

The results of both modules are collected into the *semantic object map*.

III. Robust Acquisition of Point Models

The main data structure used by our algorithms is represented by *point clouds*. The sensors produce several point clouds, each corresponding to a view or snapshot taken by the respective sensor. A *point cloud* is a set of points, each specifying an $\langle x, y, z\rangle$ position in space, possibly associated with additional information such as the brightness or the color of the respective point.

The workhorse of our object mapping approach is the *robust registration-based object reconstruction* method. It solves the following model acquisition task: given multiple views of the object to be mapped and possibly an estimation of the sensor position (not mandatory), compute a transformation that can align the views together. The algorithm outputs a polygonal reconstructed model of the object where sensor noise is largely removed based on statistical inferences and missing pieces of surface information are added.

The computational problem is difficult for various reasons. Due to the physical limitation of sensors and their limited line of sight, several "images" from multiple locations must be taken in order to obtain a complete representation of the object. The data is also noisy and often contains a substantial number of outliers. Also many objects of daily use are so complicated that self occlusions cannot be compensated by taking additional recordings. Last but not least, the computations have to be performed on huge data clouds, which prevents the use of straightforward computation techniques.

A. Overview of the Robust Acquisition

Our object mapping approach satisfies the requirements stated in the last section in that it constitutes a novel combination of computational ideas being largely independently developed in the research areas of computer graphics, robotics, and scientific computing. We combine methods for building accurate geometrical representations of 3D objects, developed by researchers in the computer graphics community [7], [8] as well as robotics [9], [10] with robust estimation techniques widely used in robotics and computer vision [11]. In order to make the computation processes feasible we apply clustering techniques and techniques from scientific computing to deal with the high volume of data.

The basic idea of the algorithm is to construct complete point models by aligning the point clouds of the corresponding object views. This is done by an iterative improvement algorithm that repeatedly rotates and positions a newly acquired point cloud P^d, such that it optimally complements another point cloud taken of the same object or scene, which we call the point model Q^d. A 50% overlap is however required between the feature points (*not* the original point sets) as defined by the robustness of the estimator. The fitness function is defined as a distance metric between points in the first scan and the corresponding points in the other scan.

This is done by the following computational steps:

1) *initialization.*
   - *noise removal.* The raw data coming from the sensors and constituting the point cloud P^d_{raw} is preprocessed using statistical techniques[12] in order to eliminate outliers and reduce noise.
     $$P^d = \{p^d_i \in P^d_{raw} \mid \overline{dist}(p^d_i, p^d_j) > \mu + d_{thresh} \cdot \sigma\} \quad (1)$$

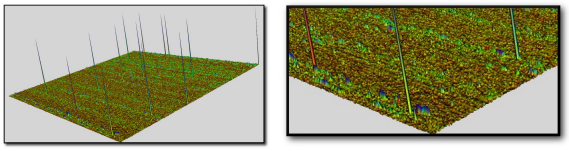

Fig. 4. Mean distances in a noisy point cloud representing a cup.

   - *computing an initial positioning hypothesis of the point cloud.* The point cloud P^d is positioned in the neighborhood of the model cloud Q^d, such that it lies within the convergence area of the algorithm. For this purpose, the position and orientation estimate of the sensor is used. If the estimate is not available or unreliable, the positioning of P^d is done based on normalizations and rotations resulting from its principal components (PCA) using eigenanalysis.
     $$\begin{cases} p_i^{'} = p_i - \frac{1}{n}\sum_{i=0}^{n} p_i, \quad q_i^{'} = q_i - \frac{1}{m}\sum_{i=0}^{m} q_i \\ C_p = P^T \cdot P \overset{\text{SVD}}{=} U_p \cdot S_p \cdot V_p^T \\ C_q = Q^T \cdot Q \overset{\text{SVD}}{=} U_q \cdot S_q \cdot V_q^T \end{cases} \quad (2)$$
     Where C_p and C_q are the covariance matrices of the de-meaned point sets. The resulting transformation is defined by: $T = Q^d - (U_q \cdot U_p^T) \cdot P^d$

2) *Determining features for optimizing the point cloud alignment.* Several geometrical (and visual - where color or intensity data is present) features are extracted and

different level of detail views are generated for the point cloud (see Figure 5).

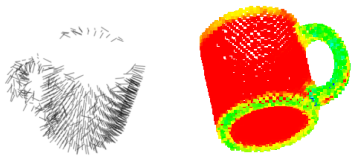

Fig. 5. Normal point surface and curvature estimation features.

3) clustering is applied to the point cloud for dimensionality reduction (a cluster is replaced by its center of mass) and faster search operations
4) a robust registration technique is applied to merge multiple views together in order to form a complete model

B. Point cloud registration

As presented above, the multiple observations gathered from the sensors need to be registered together in order to form a model which characterizes the target object. Several registration methods exist, ranging from those that compute only rigid transformations, to those applicable to deformable surfaces. In our work, we treat all objects of interest as being rigid, thus we are interested in finding out the transformation (rotation and translation) which minimizes the distance error in a least-squares sense. Specifically, we are looking for the transformation:

$$T_{(P,Q)} = \{R \cdot P + t \mid R = [3 \times 3], t = [1 \times 3]\}$$

which minimizes the error

$$e_{(R,t)} = \frac{1}{N} \sum_{i=1}^{N} \theta ||q_i - R \cdot p_i - t||^2$$

where *P* and *Q* represent the observation (point clouds), *R* is a 3x3 matrix representing the rotation, *t* a 1x3 column vector representing the translation, and θ is a weight of the current evaluated point correspondence.

One of the most widely used rigid registration algorithms is the iterative closest point (ICP) [13], [12]. Two big disadvantages of the classical ICP algorithm, besides its speed, make it unappealing for robotics: the wideness of the convergence basin (highly correlated with the needed initial alignment) and the correspondence problem (ICP assumes that each point from the source point cloud has a correspondence in the target point cloud).

We propose the use of an ICP-like algorithm which we developed: RnDICP (Robust-nD-ICP - see below). Our approach is twofold: we first process both point clouds in order to extract the most important features that could be used in the registration process (see below). Then, in the actual registration process, we try to include any additional information that we have or can extract from the sensor data, thus boosting the searching dimensionality. This means that, in comparison to the classical ICP which attempts to minimize a 3D mean distance metric between the two point clouds, our algorithm attempts to minimize a nD distance (eg. 6D for XYZ and RGB color information, 9D for XYZ, RGB and nXnYnZ normal information, etc).

The algorithm is summarized as follows:

1) Statistically remove noise and outliers from the P^d and Q^d point clouds (see Equation 1).
2) Provide an initial guess for the registration algorithm by aligning the point cloud P^d as close as possible to the model Q^d. If no position and orientation information are available from the sensor, perform eigenanalysis/PCA and use the principal components for rotation (see Equation 2).
3) Estimate geometric primitives and compute features such as surface point normals and curvature flatness metrics, etc (see Figure 5).
4) Cluster the model point cloud Q^d into k clusters $C_k = \{c_1, \ldots, c_k\}$.
5) Select the best feature points (minimal subset) from the source point cloud $P_f^d = \{p_i^d \in P^d \mid F_{p_i} > \mu + d_F \cdot \sigma\}$
6) Repeat the following steps until either the number of iterations reaches a given *maxIterations* value or the registration error drops below a given *errValue* threshold
   - For every selected feature point in cloud P_f^d, search for a corresponding point in cloud Q^d.
   $$\begin{cases} Q_n^d = \{q_{n_1}^d, \ldots, q_{n_s}^d\} \\ dist(P_f, Q_n) = min \sqrt{\sum_{i=1}^{d} \theta(P_f^i, Q_n^i)^2} \end{cases}$$
   - Statistically remove unlikely matches using the methods presented in [12].
   - Compute the rigid transformation $T = (R, t)$ using the singular value decomposition (SVD) of the covariance matrix.
   - Apply the transformation to P_f^d, and adjust the rest of the point attributes (eg. rotate normals).
   $$P_{f_j}^d = R \cdot P_{f_{j-1}^d} + t$$
   - Compute the registration error metric ϵ.
   $$\epsilon = MSE(dist(P_f^d, Q_n^d)))$$

The result of the RnDICP process will be a registered nD model, containing the X,Y,Z point coordinates together with normal information (nX,nY,nZ) as well as extra features (eg. curvature flatness [14]). Where available, R,G,B color information will be included.

Among the computed features for registration, we use the ideas presented in [14], where an edge and boundary detection algorithm is performed in order to isolate the important characteristics of the cloud from the rest. The edges of objects can be calculated from the curvature information, as they are characterized by high changes in curvature. This however won't find the boundary points in the point cloud, as there is no change in curvature for points residing on the outer border of the cloud.

Boundary points can be easily identified in 2D, as the maximal angle formed by the vectors towards the neighboring points will be larger for boundary points than for points that are on the inside of an object. In 3D however, such

angle cannot be computed, still, the method can be applied if one projects a neighborhood on a local reference plane, and transforms the coordinate system to lie on that plane. Mathematically this is the same problem as the detection of peaks in distances when removing outliers.

C. Point cloud processing and resampling

Because of noise and measurement errors during the scanning process, the model will contain outliers, or even holes, which, if left alone, would disrupt the polygonal reconstruction process. Besides the outliers, which can be detected statistically (see Equation 1), point clouds obtained with laser scanners contain small measurement errors, making a surface look *thick*. Also, after the registration process, unprecisely aligned surfaces can appear doubled.

These anomalies can be smoothed using our proposed Robust Moving Least Squares (RMLS) algorithm. The algorithm is summarized as follows: given a point cloud P^d, we want to reproduce the complete smooth surface it represents, by resampling (either upsampling or downsampling) and discarding unwanted data.

1) The coordinates are normalized using the diagonal of the point cloud's bounding box, ensuring a unity maximal distance between points. We use $h = \mu + k \cdot \sigma$ as our weight factor, where μ is the mean and σ the standard deviation of the mean distances between points
2) An initial guess is computed by approximating the point cloud with a set of points Q^3 that are in the vicinity of the original points and their coordinates are integer multiples of a specified resampling step size s:
$$Q^3 = \{q_k \in \mathbb{R}^3 \mid q_k^{xyz} = s \cdot \left| \frac{p_i^{xyz}}{s} \right|,\ p_j \in P^d\}$$
The uniqueness of the obtained points must be ensured and extra points must be added to areas where a hole was detected. The algorithm is similar to the one for boundary points detection, with the difference that here we check if a point is inside of an object, so the computed maximal angle must be sufficiently small to ensure correct filling.
To ensure that the checked point is close to the surface, its height above the reference plane should be limited with the step size.
3) For each point $q_k \in \overline{Q^3}$ to project, a local neighborhood is selected, by using either a fixed number of nearest neighbors, or all neighbors in a fixed radius. The first method can result in neighbors that are relatively far away form the currently fitted point, and their influence must be minimized when fitting the point. Therefore, the weights are assigned to every neighbor $p_j \in P^d$ based on the distance to the currently fitted point (see the equation below). The second method has the disadvantage that it may yield too few neighbors if the search radius is too small, but it ensures that the neighbors will surround the point to be fitted.
$$w_{(p_j)}^{q_k} = \exp -\frac{dist(p_j^{xyz}, q_k)^2}{h^2},\ q_k \in \overline{Q^3},\ p_j \in P_k^d$$
4) A local reference plane is selected for coordinate transformation. To fit a plane to the local neighborhood P_k^d, the minimization of the weighted squares of the error (defined as the sum of weighted distances of the points from the plane) was suggested by an iterative projection approach [15]. This method, although weighted, takes into account the whole neighborhood of the projected point, meaning that it will be influenced by a relatively small patch of outliers, which we found to be common in point clouds obtained by laser scanners. These groups of outliers cannot be eliminated statistically as in Equation 1 because they have approximately the same density as the points that are actually on the surface, so a method with a larger breakdown point is needed. We propose a weighted distance based version of the Random Sample Consensus (RANSAC) algorithm, which will yield good results for up to 50% noise in the neighborhood.
To identify the inliers $p_l \in P_{in}^d \subset P_k^d$, a random sample is selected from the neighborhood and a plane D is fitted to them. To measure the probability that the obtained plane is correct, the weighted distances of the points near the plane are summed and mapped on the $(0, 1)$ interval. Iterating through a statistically defined number of steps, the inliers will be obtained as the points close to the fitted plane. These inliers will be used as the reduced neighborhood of the projected point. It must be noted that the weighting ensures that the plane will be fitted to points that are closest to q_k. A comparison between the Weighted Least Squares and our Weighted RANSAC can be seen in Figure 6.

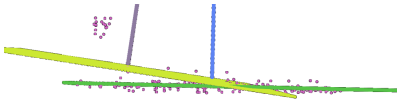

Fig. 6. WLS vs. WRANSAC (dark green) for a cloud with 10% outliers

The fitted plane's equation and the curvature can be obtained by performing eigenanalysis on the covariance matrix of the selected inliers. The color of the obtained point can be approximated with the weighted average of the inliers' colors.
$$P_g = 1 - \left(\sum_{p_j \in P_k^d} w_{(p_j)}^{q_k} \right)^{-1} \cdot \sum_{\substack{p_j \in P_k^d \\ D_{(p_j)} < dT}} w_{(p_j)}^{q_k} \cdot D_{(p_j)}$$
5) The second step of the MLS procedure is the polynomial fitting, when the xyz coordinate system is transformed into uvn, which lies on the fitted plane and a high-order bivariable polynomial is fitted to the heights of the points above the plane and the height of the fitted point is recalculated (see the equations below).
$$For\ q_k \in \overline{Q^3},\ p_l \in P_{in}^d,\ X_{(t,l)} = uniq((p_l^U)^a \cdot (p_l^V)^b$$
such that $t = \overline{1, (order+1)^2},\ a, b \in \mathbb{N}$ and $a + b < t$
$$c^{(q_k)} = (X \cdot W_{diag} \cdot X^T)^{-1} \cdot X \cdot W_{diag} \cdot p_l^N$$
where $W_{diag(l,l)}^{q_k} = w_{(p_l)}^{q_k}$

Examples of reconstructed point clouds can be seen in Figure 10.

IV. Environment Maps

After an initial model is built using the algorithm presented in Section III-A, the points lying on edges and on boundaries are segmented again from the model. This constitutes the first step towards the detection of a variety of regions of interest in the data. The results of the above algorithm can be seen in Figure 7.

We implemented the above described algorithms and tested them on data acquired using two different laser scanners. All data sets were partially overlapped, but no information regarding their orientation and position in space was used. The number of points in a data set ranged from a few thousand points to a few millions.

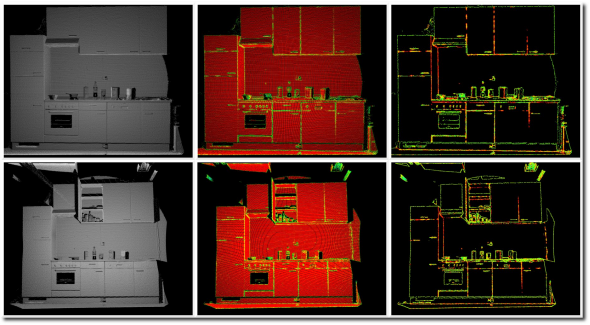

Fig. 7. Environment features (edges and boundaries) extraction

Figure 8 shows various snapshots from the environment mapping process of a kitchen. In the upper left corner, the features extracted from the two point clouds in Figure 7 are used in the robust registration process. The gray point cloud represents the target model, the blue one the source data and the red lines represent the distances between the corresponding points. The results after the registration process are shown in the upper right corner as well as in the middle picture. By subtracting both point clouds, one from the other, we obtain the results shown in the bottom part of the figure.

In the above example, the two point clouds represent two different states of the kitchen: in the first state, all cupboards are closed, while in the second state, a cupboard has been opened. This will later on be used in the segmentation process.

A. Cupboards detection

Once the edges have been extracted, the model is clustered and a search for line segments is performed, using a WRANSAC approach. In a second step, the line segments are grown into complete lines (exceeding the cluster boundaries), and a clustering is performed in the line direction space. Four lines will form a rectangle if they fulfill the following assumptions: every two of them must belong to the same class direction-wise, they must be coplanar, the angles close to 90 degrees and the rectangle edges must have a length between a minimum and a maximum given value. If duplicate rectangles that overlap are found, they will be re-segmented into smaller rectangles.

In our experiments, we have chosen the length of the rectangle edges to be between 30 and 150cm. An example of our segmentation algorithm is shown in Figure 2. By combining these results with the ones shown in Figure 8, we can match the rectangles found with the actual cupboards in the kitchen.

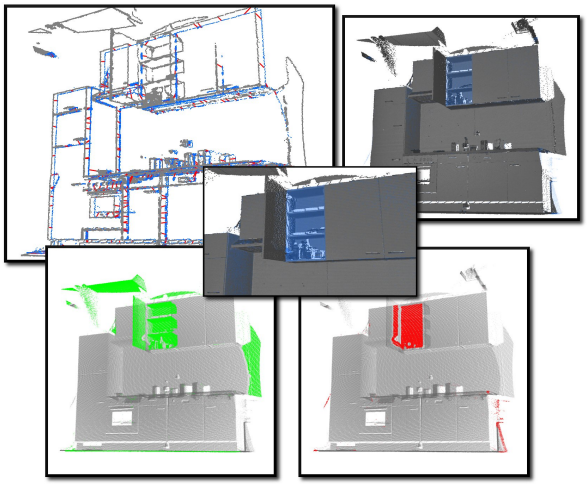

Fig. 8. The Robust-nD-ICP registration and segmentation process.

B. Tables detection

The detection of tables in a household environment has a double role. First of all, tables are interesting because they are places where objects are placed and manipulated. Secondly, by detecting tables, we can also segment and then classify the objects which sit on them.

In order to recognize what objects are placed on a table, our algorithm first attempts to detect the table itself, and then segment it out. The following assumption is taken under consideration: tables are planar areas, consisting of at least 30% of the data set's values, which are located at a height between h_{min} and h_{max}. In our experiments, we have chosen $h_{min} = 60cm$ and $h_{max} = 100cm$. The *ground plane* should be provided by the robot, using for example the kinematic data. In case this is not possible, we make the assumption that the planar area containing the largest number of points is the ground plane. The segmentation process is performed using a WRANSAC-based approach for plane detection. The results are presented in Figure 9.

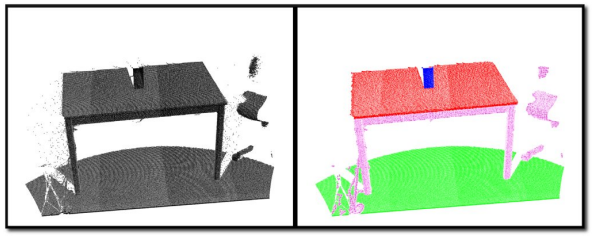

Fig. 9. Segmentation of tables and objects of interest located on them.

V. Object Modelling

Mapping the objects of daily use is different from mapping the static parts of the environment. For mapping kitchen furniture we could make simplifying assumptions, such as the existence of big rectangular planes or that it suffices to

take views of the front side. This is different for the objects of daily use. Here the shapes have much greater varieties, objects might be composed of geometric primitives, and we need models of all parts of the objects. Furthermore, the models should be orientation invariant, and if possible include features that would allow them to be recognized in a large variety of possible positions, in particular occlusions.

A. Point Cloud Reconstruction

After acquiring a point cloud model, we resample it and fill the missing holes using the RMLS algorithm described in Section III-C. Once complete, we proceed towards the polygonal reconstruction of the object, either by using constrained Delaunay triangulations or if the object is *watertight*, a version of the PowerCrust algorithm[8].

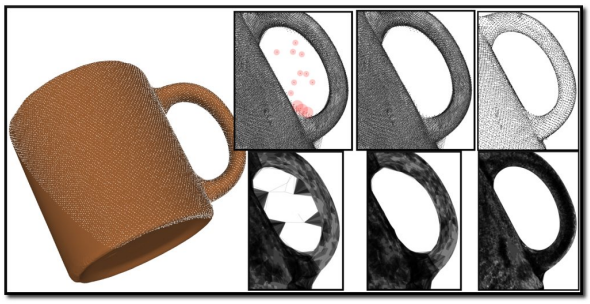

Fig. 10. Processing point clouds. From left to right in the upper part: noisy raw data acquired from the scanner, processed data with the noise removed, resampled point cloud using RMLS. In the lower part their representative polygonal reconstruction.

The resulted polygonal object will be used in the registration of a new observation taken from a sensor with the RnDICP algorithm. By computing the Mahalanobis distance between the polygonal surface and the source data set, we can hint whether the data comes from the same object or not.

Since the above method can only provide a rough estimate whether a partial scan of an object belongs to a certain model, we employ a machine learning scheme using kernel regression techniques (SVM - Support Vector Machines) in order to fully classify and differentiate between different types of objects.

In order to learn a model, the nD point cloud has to be reduced to an array of 1D features for SVM. We implemented the algorithm proposed by [9], in which four features are computed as a generalization of curvatures, based on the position and normal of each surflet and its relationship to the other surflets. For every pair of points p_1 and p_2, we define a coordinate system with the origin in the first point as $u = n_1,\ v = (p_2 - p_1) \times u,\ w = u \times v$, where n_1 and n_2 are the normals of the two points.

The four features (combined into F_c) are segmented into a specified number of bins (see Equation 3), and their histogram is used to identify the class of the object. The algorithm is orientation and position invariant.

$$\left.\begin{aligned} f_1 &= ||p_2 - p_1|| \\ f_2 &= v \cdot n_2 \\ f_3 &= atan(w \cdot n_2, u \cdot n_2) \\ f_4 &= u \cdot (p_2 - p_1)/a \end{aligned}\right\} \Rightarrow F_c = \sum_{i=1}^{i \leq 4} bin(f_i) \cdot nr_{bin}^{4-i} \quad (3)$$

The algorithm presented in [9] cannot however distinguish between subclasses of the same shape but which represent different objects (eg. a ketchup and a mustard bottle have the same shape but different colors). In this sense, we extended the algorithm by computing another histogram, this time based on the curvature and color information. As we are looking in very accentuated differences in the second histogram a very small bin size suffices.

A straightforward comparison between objects can be made directly from the histograms presented in Figure 11.

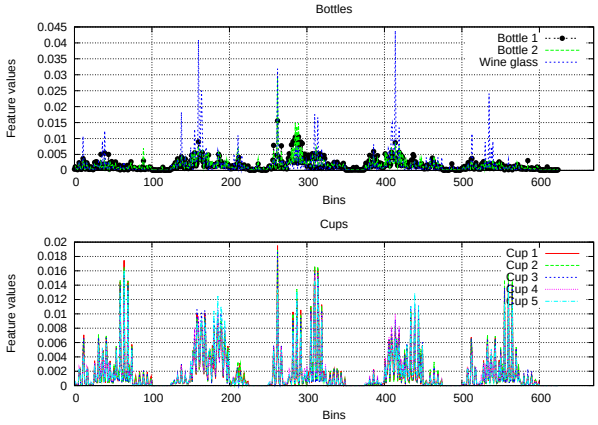

Fig. 11. Comparing different objects based on their feature histograms.

B. Learning Models

An important aspect when learning models of objects, is that for achieving a good classification accuracy, a large number of training examples is needed. Unfortunately the process of scanning objects as we discovered, is a very tedious one, and results in many hours of manual labor.

To overcome this problem, we propose the conversion and use of Google Sketchup models available online as additional training data for our application. The process of converting the models to point cloud data sets is done by the following sequence of computational steps (see Figure 13):

1) The models are mined from the Google 3D Warehouse semi-automatically using pre-defined keywords describing the objects of interest, and saved to file in a polygonal format
2) Each model is automatically processed by converting all its faces from polygons to triangles (surface triangulation split)
3) On each model, each resulted triangle is evenly fitted with points using a constant given step
4) Zero-mean Gaussian noise with various standard deviations is added to each resulted point cloud

The resulted point clouds are then processed in the same manner as the data sets acquired using the laser sensors. This

assures that plenty of training examples will be available for learning the model.

The table below describes the classification rate using three different classifiers. Our model was learned from a data set consisting of 44 training examples (eg. mugs, plates, boxes, silverware, pots, etc) and was applied to 13 new unseen objects. SVM achieved the highest classification rate.

	C4.5	NaiveBayes	SVM
Classification Rate	84.61%	69.23%	92.3%

Fig. 12. The errors of the classification process using various different learning methods: Decision Trees, NaiveBayes and Support Vector Machines

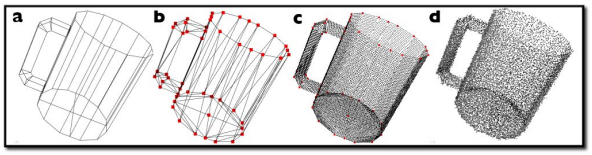

Fig. 13. Using models from the Google 3D Warehouse to improve the models: (a) polygonal data set taken saved from Google, (b) processed model containing only triangle polygons, (c) resampled point cloud model, (d) model with added zero-mean Gaussian noise

VI. Related Work

Related work can be categorized on several dimensions, but due to space constraints, we are unable to address all of them.

The Iterative Closest Point algorithm has seen several improvements over the years from its original proposed form [13], [12]. Some of them addressed the problem of computational complexity[16], selection of the best features to be used for registration[17], or including extra information in the registration process, such as colors[18] or normals[14]. An integration of all the above has not been performed yet, to the best of our knowledge.

Our Robust Moving Least Squares algorithm differs from the one described in [19], which uses LMedS, instead of RANSAC. While a comparison of both algorithms cannot be done due to the unavailability of the source codes as well as the use of different data sets, we believe that their performance should be similar. However, recent findings [20] have shown that RANSAC-like estimators do in fact perform better than LMedS ones when the data is highly contaminated with noise.

VII. Conclusion and Future work

We have presented a mapping system for acquiring models of task-relevant objects in man-made indoor environments. The significance of this work lies in the development, integration and improvements of several techniques from different research fields such as computer graphics, robotics, machine learning and scientific computing, as well as in the results presented.

We are currently investigating ways of adding more semantic information to our 3D object maps. This semantic information is obtained in two ways. First, the mapping process uses sensor data from sensor-equipped indoor environments. Second, we couple the map acquisition process to a activity recognition system. This way we obtain additional information about the role of the map objects in activities. Taken together we consider these steps to be critical for the development of semantic object maps.

VIII. Acknowledgments

This work is supported by the CoTeSys (Cognition for Technical Systems) cluster of excellence. We would like to thank our colleagues Suat Gedikli and Philipp Kemmeter for their ideas and contributions to the project.

References

[1] S. Engelson and D. McDermott, "Maps considered as adaptive planning resources," in *Fall Symposium on App. of Artificial Intelligence to Real-World Autonomous Mobile Robots*, 1992, pp. 36–44.

[2] S. Vasudevan, S. Gachter, V. Nguyen, and R. Siegwart, "Cognitive maps for mobile robots - an object based approach," 2007.

[3] 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.

[4] 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.

[5] 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 (GfKl)*, Freiburg, Germany, 2007.

[6] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2002, to appear.

[7] T. K. Dey and S. Goswami, "Tight cocone: a water-tight surface reconstructor," in *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, 2003, pp. 127–134.

[8] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust," in *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*. New York, NY, USA: ACM Press, 2001, pp. 249–266.

[9] T. Bodenmueller and G. Hirzinger, "Online Surface Reconstruction from Unorganized 3D-Points for the DLR Hand-Guided Scanner System," in *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission (3DPVT'04)*, 2004.

[10] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun, "Using EM to Learn 3D Models with Mobile Robots," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[11] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," in *Comm. of the ACM, Vol 24, 1981*.

[12] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves, Tech. Rep. RR-1658.

[13] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, 1992.

[14] K.-H. Bae and D. D. Lichti, "Automated registration of unorganized point clouds from terrestrial laser scanners," in *International Archives of Photogrammetry and Remote Sensing (IAPRS)*, 2004, pp. 222–227.

[15] A. M. and A. A., "On normals and projection operators for surfaces defined by point sets," in *Proceedings of Symposium on Point-Based Graphics 04*, 2004, pp. 149–155.

[16] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 145–152, 2001.

[17] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust Global Registration," in *Proc. Symp. Geom. Processing*, 2005.

[18] A. E. Johnson and S. B. Kang, "Registration and integration of textured 3-D data," in *NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*. Washington, DC, USA: IEEE Computer Society, 1997, p. 234.

[19] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, 2005, pp. 544–552.

[20] L. Hajder and D. Chetverikov, "Weak-perspective structure from motion for strongly contaminated data," *Pattern Recogn. Lett.*, vol. 27, no. 14, pp. 1581–1589, 2006.