Robotics and Autonomous Systems 56 (2008) 927–941

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Towards 3D Point cloud based object maps for household environments

Radu Bogdan Rusu *, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, Michael Beetz

Technische Universität München, Computer Science Department, Intelligent Autonomous Systems Group, Boltzmannstr. 3, 85748, Garching bei München, Germany

ARTICLE INFO

Article history:
Available online 20 August 2008

Keywords:
Environment object model
Point cloud data
Geometrical reasoning

ABSTRACT

This article investigates the problem of acquiring 3D object maps of indoor household environments, in particular kitchens. The objects modeled in these maps include cupboards, tables, drawers and shelves, which are of particular importance for a household robotic assistant. Our mapping approach is based on PCD (point cloud data) representations. Sophisticated interpretation methods operating on these representations eliminate noise and resample the data without deleting the important details, and interpret the improved point clouds in terms of rectangular planes and 3D geometric shapes. We detail the steps of our mapping approach and explain the key techniques that make it work. The novel techniques include statistical analysis, persistent histogram features estimation that allows for a consistent registration, resampling with additional robust fitting techniques, and segmentation of the environment into meaningful regions.

1. Introduction

In this article we investigate how an autonomous mobile robot (see Fig. 1) can acquire an environment object model of a kitchen, and more generally of human living environments. The acquired environment model is to serve as a resource for robotic assistants that are to perform household chores including setting the table, cleaning up, preparing meals, etc. For tasks like these, it is not sufficient to have models that serve localization and navigation purposes. Rather the robot has to know about cupboards, tables, drawers, shelves, etc, how to open and use them, what is stored in them, etc. Finally, the models have to represent ovens, dish washers and other appliances as specific kinds of cupboards.

Our scenario is as follows: a robot enters the room and repeatedly turns and makes a sweeping movement with its left arm that has a a laser sensor mounted on its end effector (see Fig. 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 $\langle x, y, z \rangle$, intensity and distance values $\langle i, d \rangle$. After processing, integrating, and interpreting the individual scans, the robot transforms the individual point clouds into an environment object model (simply called *object map*) as it is depicted in Fig. 2. This object map consists of a 3D mesh visually representing the parts of the environment that the robot considers as obstacles that it must not collide with, and cuboids and planes that represent the objects that are relevant for the robot's tasks — appliances, cupboards, drawers, and tables.

We represent the object map using OWL-DL[1] such that the environment models can be easily shared between different applications and robots, and that a robot may query the map in order to retrieve necessary information from the model. The objects in our map are hierarchically structured, such that kitchenettes consist of connected cupboards, a single cupboard is a box with a door and a handle. The objects are also 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. Or, an oven is a specific kind of container that is used to prepare meals.

The main challenge of this scenario is the transformation of raw point cloud data into a meaningful, compact representation of the world, without losing fine grained details necessary for tasks like robot manipulation and activity recognition. The key contributions of this article include a novel multi-dimensional tuple representation for point clouds and robust, efficient and accurate techniques for computing these representations, which facilitate the creation of hierarchical object models for kitchen environments.

The tuples contain informative point feature histograms, persistent feature values, and region connectivity information, among other information. The techniques which operate on them include: a robust and fast registration using persistent feature histograms, robust polynomial resampling methods, and segmentation and extraction of objects in the map.

* Corresponding author. Tel.: +49 08928917780; fax: +49 08928917757.
E-mail addresses: rusu@cs.tum.edu (R.B. Rusu), marton@cs.tum.edu (Z.C. Marton), blodow@cs.tum.edu (N. Blodow), dolha@cs.tum.edu (M. Dolha), beetz@cs.tum.edu (M. Beetz).

[1] We use ResearchCyc as our basic encyclopedic knowledge base and add the object classes that are missing and needed for representing our maps. ResearchCyc knowledge bases can be exported into OWL-DL, which makes them usable as Semantic Web knowledge sources, and applicable to our map representation.

0921-8890/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2008.08.005

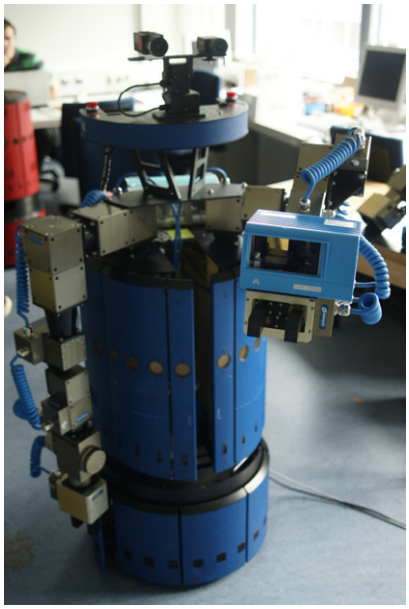

Fig. 1. Autonomous robot used for the experiments in this article. A laser scanner is mounted on the arm, which allows the robot to acquire 3D scans of the environment.

We show the effectiveness of the contributions, and their usage in a comprehensive application scenario. This includes constructing a complete point cloud model of the environment, the extraction of informative features for geometrical reasoning, segmentation into classes of objects, and supporting descriptive queries useful in high level action planning.

The remainder of the article is organized as follows. Section 2 presents related work, followed by a description of our system overview and architecture in Section 3. Sections 4 and 5 explain the technicalities of the two main modules of our system, namely Geometrical Mapping and Functional Mapping. We briefly reiterate our empirical results in Section 6, and conclude with Section 7.

2. Related work

Several efforts in the past have been made regarding the creation of environment 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 [18]. The maps are created using mobile robots equipped with laser range finders, but they do not include any semantic information. The work in [13] 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 [8] 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 [38]. The work presented in [21] 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 [24] 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). With few exceptions, in particular in the area of cognitive mapping [38,20], but also including [37], maps do not represent objects relevant for other robot tasks, besides navigation.

Since the input data leading to the creation of the object map has to be acquired in stages, an algorithm for automatically aligning (registering) the separate views into the same model is needed. One of the most popular registration methods to date is the Iterative Closest Point (ICP) algorithm [5,41]. The original version uses pairs of nearest 3D points in the source and model set as correspondences, and assumes that every point has a corresponding match. Obviously this is rarely the case with most applications, but a simple distance threshold can be used to disregard correspondences in order to deal with partially overlapping scenes. Additionally, to further improve the quality of correspondences, a lot of efforts have been made into the area of feature selection [10,35], as well as including extra information such as colors [17] or normals [2] that could improve the correspondence problem. Since ICP requires an exhaustive search through the correspondence space, several variants that address the problem of its computational complexity have been proposed [28,22]. Furthermore, methods for generating initial approximate alignments [10,35,19], as well as choosing alternate optimization methods [14,6], improved the results of the registration process. [23] presents a system for 3D laser scan registration using ICP based on semantic information (e.g. walls, floors, ceilings), which decreases the computation time by up to 30%. Our registration algorithm is based upon the previously mentioned work, and adds several extensions. In the following we will discuss our reasons and justify the need for these extensions.

While feature candidates such as surface curvature estimation [2] or integral volume descriptors [10] have already been used

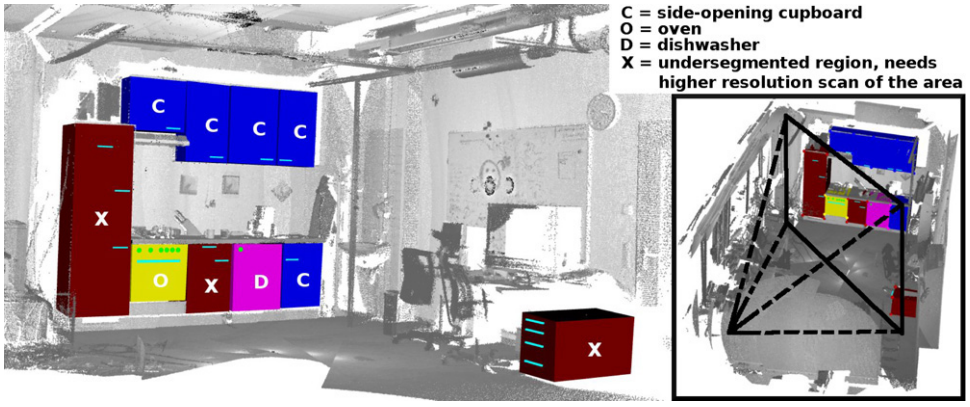

Fig. 2. An object map of the kitchen environment with superimposed object classes (remission values shown in grayscale).

as features for finding better matches in the point-to-point correspondence process, these only represent their point neighborhoods partially, and are dependent on noise levels or point cloud density. In general, descriptors that characterize points with a single value might not be expressive enough to make the necessary distinctions between points lying on different geometric surfaces. As a direct consequence, most scenes will contain many points with the same or very similar feature values, thus reducing their informative characteristics. Multiple-value point features such as curvature maps [9], or spin images [16], are some of the better local characterizations proposed for 3D meshes which got adopted for point cloud data. These representations require densely sampled data and are unable to deal with the noise usually present in 2.5D scans.

We propose the use of a better system that combines several aspects of the geometry of a point's neighborhood for feature estimation [39], which already showed promising results in our previous work [29,32,30], as a *multi-value* feature set for selecting point correspondences.

A system able to compute a rough initial guess for the ICP algorithm, using Extended Gaussian Images (EGI) and the rotational Fourier transform is presented in [19]. Extended Gaussian Images are useful for representing the shapes of surfaces and can be approximated using the spherical histogram of surface orientations. While this can provide good transformations which might align two datasets closely for watertight objects, it does not produce satisfactory results for our environment models, as the normals alone do not provide enough information about the geometry of the cloud. Our work is based on the idea that relationships between persistent features in a scene can lead to potentially better alignment candidates. Therefore we assemble pairs of histogram-features in the first point cloud and check for corresponding pairs (points with similar histograms and distances) in the second one, similar to [10].

Once a complete PCD model is obtained further segmentation and geometrical reasoning algorithms need to be applied, to extract useful information from it. MLS (Moving Least Squares) is a widely used technique for approximating scattered data using smooth functions. [1] describes a method to fit polynomials to a point cloud based only on coordinate data and estimated surface normals. To counteract the effects of noise, our approach uses Sample Consensus like methods and other robust estimators in order to calculate the best reference plane before fitting a high order polynomial [29]. Another Robust Moving Least Squares algorithm is described in [7], which differs from ours in that it uses LMedS instead of our sample consensus (SAC) based model fitting approach. To obtain a polygonal model of the world, further surface reconstruction methods can be applied such as [15,12]. In our work, we use an algorithm similar to [12], with the exception that we don't create an additional voxel structure, but use the raw point cloud data to create the mesh. Segmenting a point cloud into regions of interest has been performed in different contexts, based on edges [33], curvature estimates [4], and smoothness constraints [27]. An automatic algorithm to detect basic shapes in unorganized point clouds is presented in [34]. The method is based on random sampling and detects planes, spheres, cylinders, cones and tori. Since we are mostly interested in cuboids for cupboards detection, the method is not directly applicable in our scenario, but this topic is still under investigation.

3. System overview

Given the scenario laid out in the Introduction we are investigating the following computational problem: from a set of point clouds, each resulting from 3D laser scans where the scans have enough overlap and cover a substantial part of a room in a human apartment, say a kitchen, automatically compute an integrated mesh representation of the individual point clouds where cupboards and drawers are represented as cuboid containers with front doors and handles, whereas tables and shelves are represented as horizontal planes.

The computational process that solves this problem is organized as depicted in Fig. 3. The raw data that constitutes the input to our system are the individual point clouds resulting from an arm sweep with the hand mounted laser scanner.

Using a series of processing steps, the Geometrical Mapping module transforms the PCD view into an interpreted PCD view. The interpreted PCD view is an improved point cloud, with uniformly resampled 3D coordinates, partially noiseless, and additional computed features that include the surface curvature and normal estimates at each point, and a characterization of the geometric form of the local point neighborhood using feature histograms. The relevance of the features is assessed by admitting features that are persistent over different resolutions. The interpreted PCD view constitutes the basic data structure for the integration of individual views in globally consistent models and for the recognition of objects in the point clouds.

The next step in the environment object model acquisition process is the registration of individual interpreted PCD views into an integrated and consistent global point cloud model of the world, correcting the imprecise alignment caused by odometry, which we call a *PCD world model*. After registration, the complete model is resampled again to account for registration errors and point density variations.

As part of the Functional Mapping module, object hypotheses are generated and included as individual entities in the map, and rectangular planes and boxes of specified size ranges will be further assessed as candidates for cupboards and tables. Finally, a Functional Reasoning step takes the PCD world model and the object hypotheses generated in the previous step as its input and transforms it into a mesh representation of the environment extended with specific object models as depicted in Fig. 2. To compute this object-based environment model the robot monitors acquired PCD views over time.

In the remainder of this article we will explain the technicalities of these computational steps and demonstrate the operation of our system by assessing the results of a complete model acquisition process.

4. Geometrical mapping

The overall architecture of the Geometrical Mapping module and the pipeline of its processing steps is presented in Fig. 4. These steps result in an interpreted PCD view that includes persistent feature histograms, normal and surface curvature estimations, and 3D resampled point coordinates. The interpreted PCD views will be registered to form a complete PCD world model.

Many geometric properties of points in a PCD are defined in terms of a local neighborhood around a query point rather than a single point. A common approach for selecting the neighborhood is to use spatial decomposition techniques (e.g. a kD-tree), and search for the point's closest neighbors using a 3D Euclidean distance as a metric. The search can be performed either until k points have been found, or until all the points confined within a bounding sphere of radius r have been selected. The k and r variables depend on how local we want the selection to be. As our point clouds were obtained by scanning real world objects, the selection of the local neighborhood can be correlated to the known scale and resolution.

4.1. Sparse outlier removal

Laser scans typically generate PCDs of varying point densities. Additionally, measurement errors lead to sparse outliers which

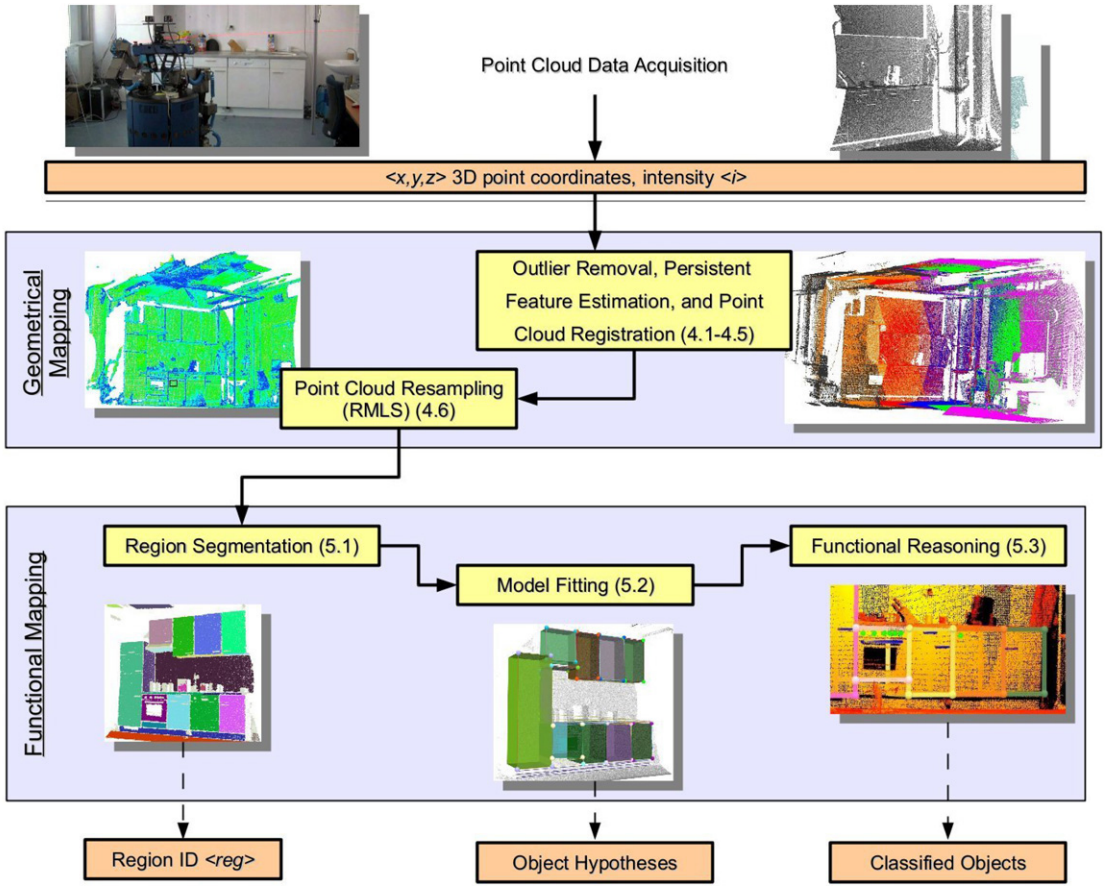

Fig. 3. The computational structure of the acquisition of object maps.

corrupt the results even more (see Fig. 5). This complicates the estimation of local point cloud characteristics such as surface normals or curvature changes, leading to erroneous values, which in turn might cause point cloud registration failures. The sparse outlier removal module corrects these irregularities by computing the mean μ and standard deviation σ of nearest neighbor distances, and trimming the points which fall outside the $\mu \pm \alpha \cdot \sigma$. The value of α depends on the size of the analyzed neighborhood.

In our implementation, we set $\alpha = 1$ and $k = 30$, because experiments with multiple datasets have confirmed the applicability of the $\mu \pm \sigma$ thresholds, with approximatively 1% of the points being considered to be noise (see Fig. 5).

4.2. Normal and curvature estimation

The normal and curvature estimation module computes an approximation of the surface's normal and curvature at each point from the PCD, based on its relationships with the nearby k points surrounding it. This information is then used for computing persistent features and registration. Determining these local characteristics fast and accurately requires: (i) a method for determining the best k-neighborhood support for the query point; and (ii) a way of estimating the surface normal and the curvature at the query point.

The estimated normal n of the point p can be approximated with the normal to the k-neighborhood surface by performing PCA (Principal Component Analysis) on the neighborhood's covariance matrix [25]. The eigenvector corresponding to the smallest eigenvalue gives an estimate of n's direction. We use a MLESAC (Maximum Likelihood Estimation SAmple Consensus) [36] technique to robustly estimate the best support for a plane and discard the outliers.

After finding the best model for point p, we assemble a weighted covariance matrix from the points p_i of the support neighborhood (where $i = 1 \ldots k$):

$$C = \sum_{i=1}^{k} \xi_i \cdot (p_i - \overline{p})^{\mathrm{T}} \cdot (p_i - \overline{p}), \quad \overline{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i \tag{1}$$

followed by the eigenvector V and eigenvalue λ computation $C \cdot V = \lambda \cdot V$. The term ξ_i represents the weight for point p_i: $\xi_i = \exp(-\frac{d_i^2}{\mu^2})$, if p_i outlier, and $\xi_i = 1$, if p_i inlier — where μ is the mean distance from the query point p to all its neighbors p_i, and d_i is the distance from point p to a neighbor p_i. Our method has the advantage that it takes into account the distance from every outlier (found using the robust estimator above) to the query point, thus changing the model slightly. This will guarantee that correct normals are estimated even in the proximity of sharp edges.

The surface curvature estimate at point p_i is defined by [25]: $\gamma_p = \lambda_0/(\lambda_0 + \lambda_1 + \lambda_2)$ where $\lambda_0 \leq \lambda_1 \leq \lambda_2$ are the eigenvalues of the covariance matrix C. Fig. 6 presents the effect of the k-neighborhood support selection in a noisy point cloud for surface curvature estimation, using standard techniques (left), and using our method (right). Because in our method the surface curvature is estimated based on a better k neighborhood support, edges are much sharper, which aids segmentation.

Due to the lack of surrounding neighboring points for query points lying at the boundaries of a region, their estimated surface curvatures as well as their normals will be computed erroneously. It is therefore important to identify such points and mark them appropriately in the point cloud [29].

The estimated point normals give a good approximation of the surface normals, but the results obtained using PCA are in general

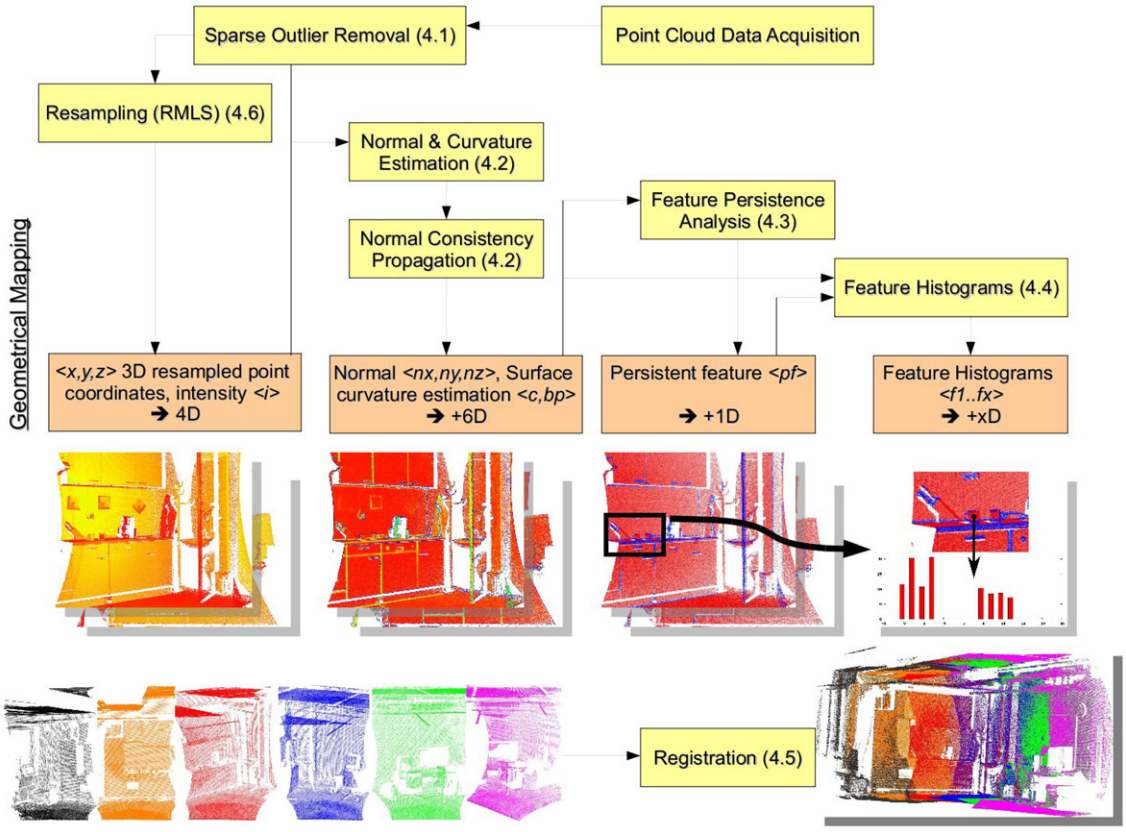

Fig. 4. The architecture of the Geometrical Mapping module.

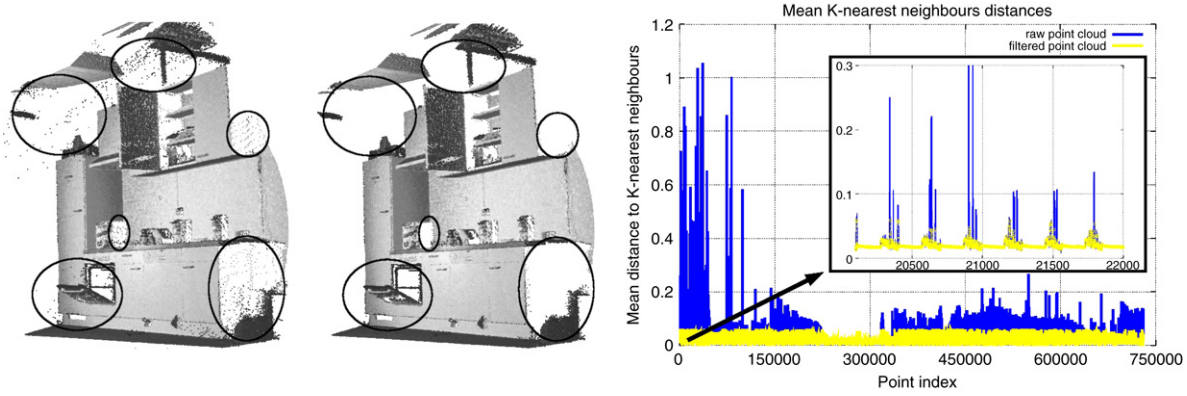

Fig. 5. Left: raw scan; middle: scan after applying gross outlier removal (points left: 728 070 out of 730 664, i.e. 99.64%); right: mean distances to $k = 30$ neighbors before and after removal using $\alpha = 1$. In both scans remission values are shown in grayscale.

not consistently oriented (see Fig. 7). However, taking advantage of the known viewpoint v we can flip the normals that aren't pointing towards it. To do so, we flip all normals n_{p_i} that form an angle $\theta > 90^\circ$ with v.

The right part of Fig. 7 shows the EGI (Extended Gaussian Image, i.e. normal sphere) and the orientation of point normals directly after PCA (top), as well as the results after re-orienting them consistently towards the viewpoint (bottom). Note how normals are dispersed on both parts of the EGI before propagating the consistent orientation.

4.3. Feature persistence analysis

When using point features as characteristics of the entire point cloud, it's good to make sure that we find a compact subset of points that best represents the point cloud. The lesser the number of feature points and the better they approximate the data, the more efficient is the subsequent interpretation process.

In order to select the best feature points for a given cloud, we analyze each point p multiple times. We vary the radius of the neighborhood's bounding sphere r over an interval depending on the point cloud size and density ($i = 1 \ldots m$), and compute different values for a feature, say γ_p for each r_i. We compute the mean μ_i and standard deviation σ_i of the feature distribution, and compute P_{f_i} as the set of those salient points whose features are outside the interval $\mu_i \pm \sigma_i$ (see Fig. 8). Finally, P_f is:

$$P_f = \bigcup_{i=1}^{m-1} (P_{f_i} \cap P_{f_{i+1}}). \tag{2}$$

Fig. 9 shows the results, with the final distribution and selected persistence features marked as *pf* for 4 different radii $r_{1,2,3,4}$. The

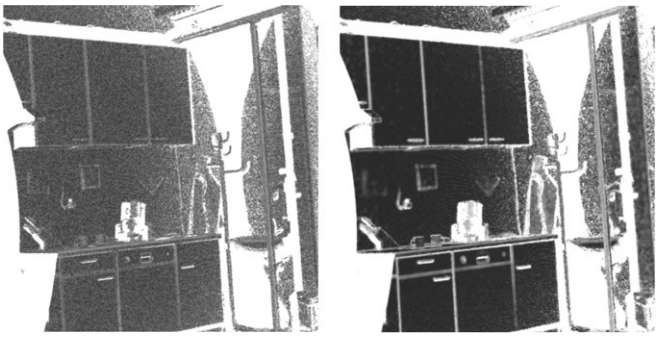

Fig. 6. Estimating surface curvatures for a noisy point cloud: using all $k = 30$ neighbors as inliers (left), and selecting automatically the best k-neighborhood support using our method (right). In both scans, curvature values are shown in grayscale: low curvatures are darker.

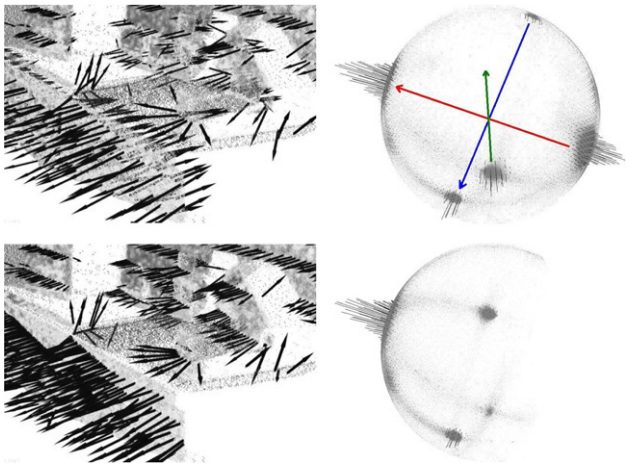

Fig. 7. Normal orientations in a point cloud and their respective EGIs: before normal flipping (top row) and after (bottom row).

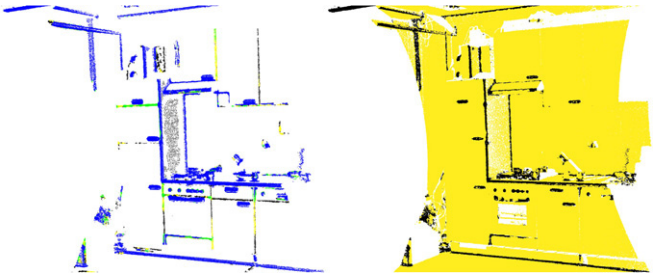

Fig. 9. The computed persistent features for each radius separately (r_1 — black, r_2 — yellow, r_3 — green, r_4 — blue) on the left, and the global persistent features (in black) over all radii in the original dataset (in yellow) on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

values of the r_i radii set are selected based on dimensionality of the features that need to be detected. Because small fine details are needed in our work at the expense of more data, (i.e. gaps between cupboard doors), we have selected $r_{\min} = 1.5$ cm and $r_{\max} = 4$ cm.

4.4. Feature histograms

While surface curvature changes or integral volume descriptors [10] can be considered as good point features for some problems, they do not always represent enough information for characterizing a point, in the sense that they approximate a k-neighborhood of a point p with a single value. As a direct consequence, further processing steps such as registration will deal with multiple and false correspondences. Ideally, we would like to have point labels such as: point on an edge, point on a sphere or a plane, etc. At the same time, the features must be fast to compute.

In order to efficiently obtain informative features, we propose the computation and usage of a histogram of values [29,30,32] which approximates the neighborhood much better, and provides an overall scale and pose invariant multi-value feature. The histogram is computed using four geometric features as proposed in [39], and generalizes the mean surface curvature at a point p_i. To reduce the computational complexity, we select only $p_i \in P_f$.

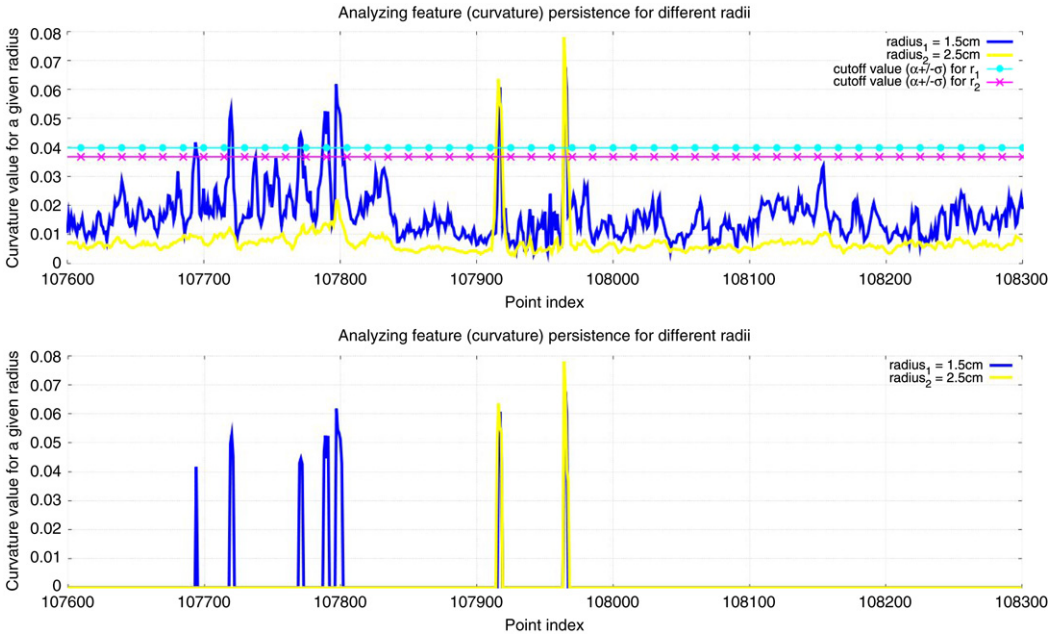

Fig. 8. Selecting the salient points based on the distinctiveness of their curvatures.

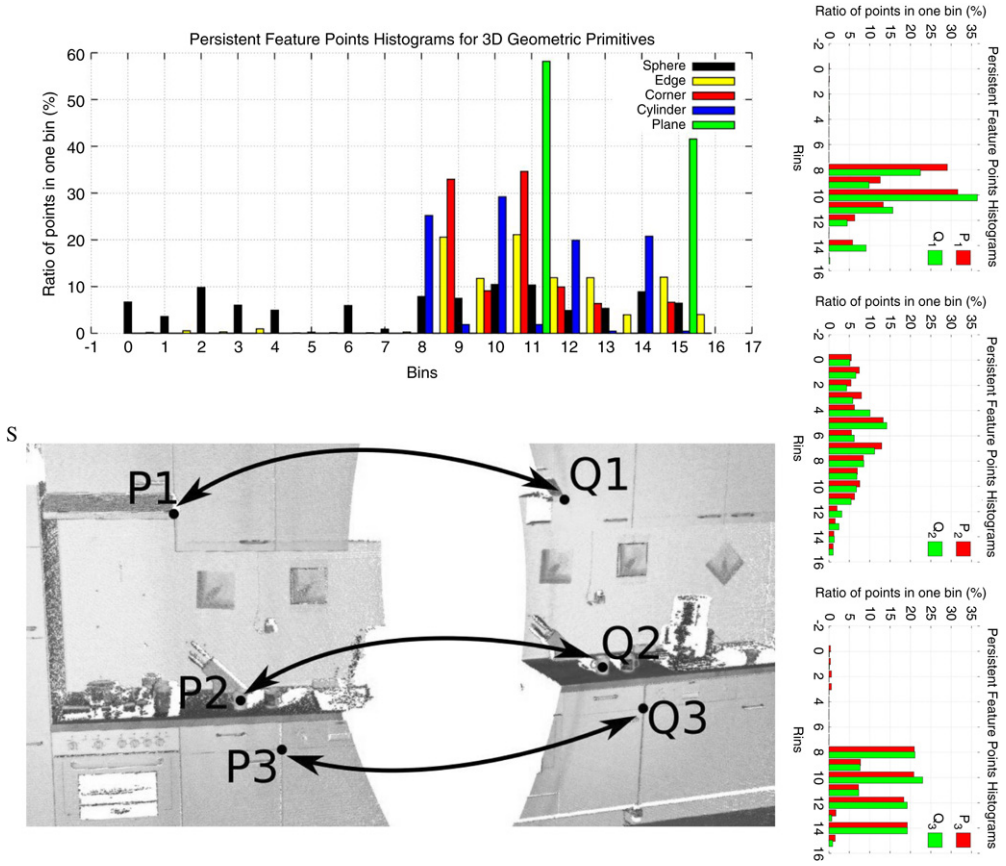

Fig. 10. Top left: Feature Histograms for query points located on different geometric surfaces; bottom and right: feature histograms for corresponding points on different point cloud datasets (remission values shown in grayscale).

For each pair of neighboring points p_{j_1} and p_{j_2} ($j_1 < k, j_1 \neq j_2$, $j_2 < j_1$) of p_i, and their estimated normals n_{j_1} and n_{j_2}, we define the Darboux frame with the origin in
p_i: $u = n_s$, $v = (p_t - p_s) \times u / \|p_j - p_i\|$, $w = u \times v$.

We then compute 4 features f_x where $x \in \{0, 1, 2, 3\}$ that measure the angle differences between the points' normals and the distance vector between them, and bin the values into a histogram (see [32] for a thorough explanation and analysis of the four histogram features chosen).

For each feature f_x we define its theoretical definition range: $(f_{x_{\min}}, f_{x_{\max}})$, and divide it in d subdivisions labeled with $0, 1, 2, \ldots, d-1$. Therefore, for a given feature value f_x, we can find out the subdivision label in which it is located, by computing the integer part operation on its value's proportion in its definition interval. The combination of all four subdivision labels yields an index in the histogram i_{hist} in which the point pair falls.

$$\left.\begin{array}{l} f_0 = \langle v, n_j \rangle \\ f_1 = \langle u, p_j - p_i \rangle / \|p_j - p_i\| \\ f_2 = \|p_j - p_i\| \\ f_3 = a\tan(\langle w, n_j \rangle, \langle u, n_j \rangle) \end{array}\right\} i_{\text{hist}} = \sum_{x=0}^{x \leq 3} \left\lfloor \frac{f_x \cdot d}{f_{x_{\max}} - f_{x_{\min}}} \right\rfloor \cdot d^x \qquad (3)$$

where the $\langle\ \rangle$-operator denotes the scalar product, and $\lfloor\ \rfloor$ the floor function. For each point-pair and its i_{hist} index, we increment the histogram value at that index by 1, and at the end, normalize each bin with the total number of point pairs $k(k+1)/2$ to achieve point density invariance.

The number of histogram bins that can be formed using these four geometric features is d^4. For example, by dividing the feature definition range in 2 parts (smaller or greater than $(f_{x_{\max}} - f_{x_{\min}})/2$, we obtain a total of $2^4 = 16$ bins as the total number of combinations between the 4 features. Because the number of bins increases exponentially by the power of 4, using more than 2 subdivisions would result in a large number of extra dimensions for each point (e.g. $3^4 = 81$D), which makes the computational problem intractable.

The top left part of Fig. 10 illustrates the differences using our proposed 16D feature set between query points located on various geometric surfaces. The surfaces were synthetically generated to have similar densities and scales as our input datasets. For each of the mentioned surfaces, a point was selected such that it lies: (i) on the corner of a cube, (ii) on the middle of an edge of a cube, (iii) on a side of a cylinder at half the height, (iv) anywhere on a sphere, and (v) on a plane; and the 16D feature histogram was generated using all its neighbors inside a sphere with radius $r = 2$ cm. The results show that the different geometrical properties of each surface produce unique signatures in the feature histograms space.

Since the point feature histograms are pose and scale invariant, they prove to be more suitable candidates for problems like correspondence search while registering multiple scans under the same model. Fig. 10 presents corresponding histogram features for similar points in two different overlapping point cloud datasets.

4.5. Registration

Once a set of interpreted PCD views has been acquired and processed, they need to be aligned together into a single point cloud model for further segmentation and object recognition. The registration module makes use of the list of persistent feature histograms from each interpreted PCD view, to compute the best rigid transformation that minimizes a distance metric error between the datasets.

To register the partial views, we use a robust variant of the popular ICP (Iterative Closest Point) algorithm [5,41]. ICP is a

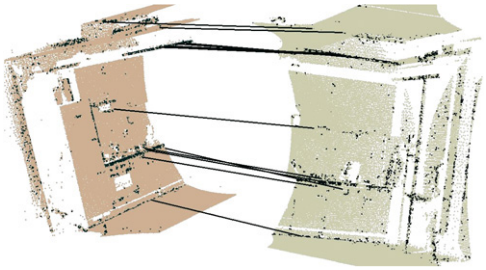

Fig. 11. Correspondences found using our method for initial alignment.

widely used algorithm for point cloud registration, but it has certain drawbacks, such as a small convergence basin, and a high number of needed iteration steps until convergence is reached. Our contributions to make ICP more robust include the use of persistent feature histograms in computing an initial alignment step to bring two point clouds into the convergence basin of ICP, and the adaption of a point-to-surface registration algorithm using instantaneous kinematics [26] to the point-to-point registration problem we are facing in our application.

The registration problem becomes easily solvable if the point to point correspondences are perfectly known in both datasets. Since that is not the case in our application, it is essential to robustly estimate good correspondences. Thus, the first step of our registration module is to compute a good initial alignment, and return a set of m correspondences that can be used to directly transform the source point cloud into the convergence area of the target. The method is similar to [10], but instead of using Integral Volume Descriptors (IVD) to identify interesting feature points, we use our 16D feature histograms which yield better results for our datasets, as show in Fig. 12. The computed correspondences must rely on the intrinsic, rotation and position invariant properties of the point cloud. Point to point distances within the point clouds are a good candidate since they are easy to compute and fulfill this requirement, so we identify a set of m points p_i that have the same distances to each other than their corresponding points q_i.

In the first step of the initial alignment, we find a set of corresponding points in the target point cloud for the persistent features of the source. To measure the similarity of two points, we compute the distance of the histograms using an appropriate distance metric as described in detail in [30,32]. In order to do this efficiently, we construct a 16 dimensional kD-tree in the feature histograms space, and for each persistent feature point in the source dataset we perform a k-nearest neighbor search in the target. Hence, we look at the k points with most similar histograms and keep them as correspondence candidates for p_i. We call this set of correspondence candidates $C = \{c_i \parallel c_i = \langle p_i, q_{i1}, \ldots, q_{ik} \rangle, 0 < i < m\}$.

In the second phase of the initial alignment, we hierarchically merge the best entries c_i^2, c_j^2 from C_2 into 4-point correspondences c_i^4 to build a set of 4-point correspondences C_4, and continue generating sets C_{2k} from C_k until there are not enough correspondences left in C_k to generate $2k$-point correspondences. We show the 16 correspondences found with this method for our kitchen dataset in Fig. 11. After an initial alignment has been obtained from this last set of correspondences, in order to improve convergence speed the algorithm uses a different method to estimate the optimal transform than classical ICP. We are using an alternative error function that instead of minimizing the sum of squares of distances between corresponding points: $\min \sum_{i=1}^{n} \text{dist}(R \cdot p_i + T, q_i)$, uses an approximate measure of point-to-surface distances [26].

Using the new distance measure, the classical ICP error function:

$$\sum_{i=1}^{n} \|R \cdot p_i + T - q_i\|^2 \text{ changes to } \sum_{i=1}^{n} \|(R \cdot p_i + T - q_i) \cdot n_{q_i}\|,$$

where n_{q_i} is the normal to the surface in point q_i. This means we try to minimize the distance between a point p_i and the surface of its corresponding point q_i, or the distance along the normal of q_i. Specifically, movement of the source point cloud tangentially to the target is not restricted. This means faster convergence when the point clouds are already close to each other, i.e. in the last iteration steps.

We conducted several experiments to evaluate our registration method. Computing the initial alignment using feature histograms proved to be far superior to using other features such as IVD or surface curvature estimates, since it was able to robustly bring the point clouds close to the correct position independent of their original poses (see Fig. 12). We could only achieve this using IVD when we considered at least $k = 150$ similar points in the target point cloud for every selected feature point in the source, and even then there were cases where it failed. Also, runtime increases exponentially in k, which also renders the IVD method inferior. We empirically discovered that $k = 10$ is a good value for the number of correspondent candidates, as increasing it further did not improve the results.

In the second stage of registration, our variant of ICP using instantaneous kinematics, converges faster to the correct solution than regular ICP. This is mostly due to the point-to-surface error metric which doesn't restrict tangential movement, thus accelerating convergence when the point clouds are already close to each other. This has no closed-form solution, but its approximation has better convergence properties than regular ICP [26] (quadratic vs linear convergence). Fig. 13 presents 5 successfully aligned PCD views and the registration errors for the first two views, after each iteration step, using the two ICP variants. The error units are degrees for the rotational error obtained from the axis-angle representation of the rotation matrix and meters for the translational error. Using histogram feature correspondences, Kinematic ICP was able to converge in 1–2 steps as opposed to at least 5 steps in Regular ICP, which also converged to a

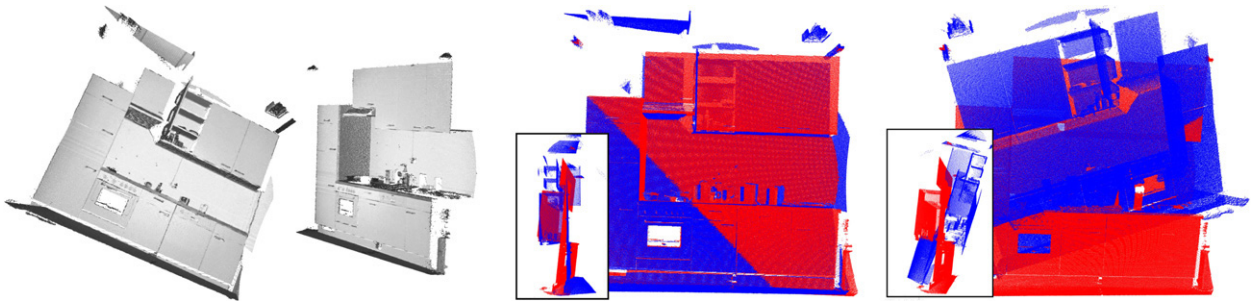

Fig. 12. Left: two datasets before registration (remission values shown in grayscale). Middle and right: Initial Alignment results using Feature Histograms and using IVD.

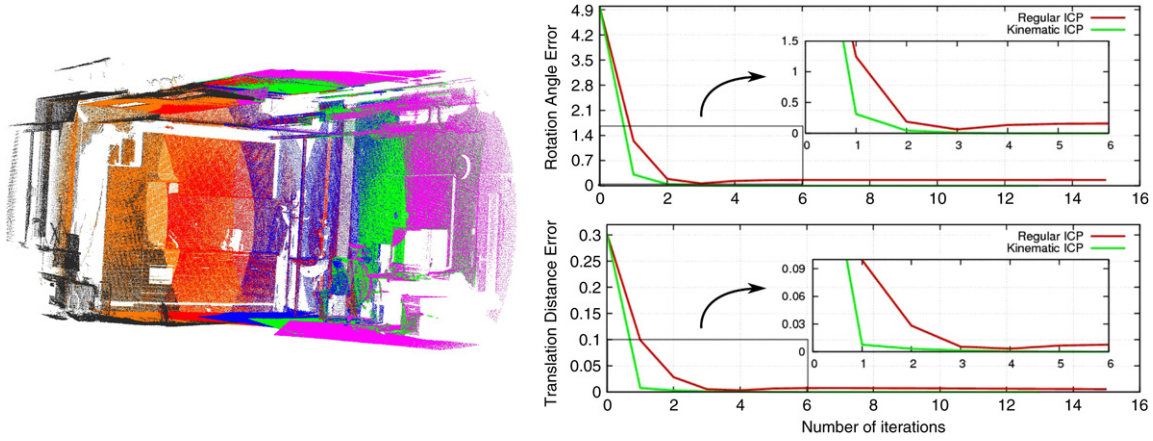

Fig. 13. Successful registration of 5 scans (left); Comparison of the registration errors of Kinematic vs. Regular ICP during each iteration step, both starting after our Initial Alignment (right). The error metric is computed against the converged solution (verified by ground truth manual measurements).

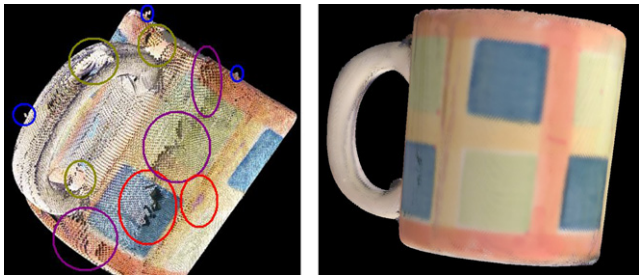

Fig. 14. Registered PCD model with: measurement errors (red), registration errors (purple), holes (brown), outliers (blue) and noise; Results after resampling (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

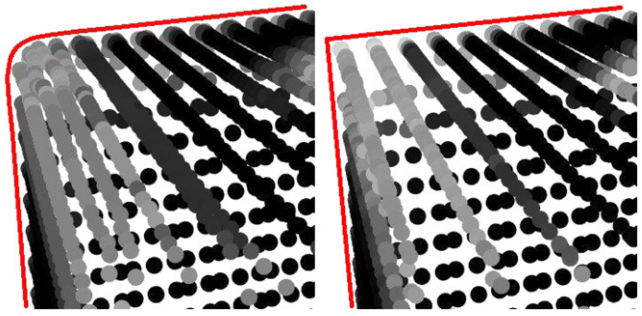

Fig. 15. Comparison of edge/corner preservation between standard MLS (left) and our Robust MLS (right) for a cube. Both pictures show curvature values in grayscale: low curvatures are darker.

slightly offset solution. The correct solution was selected as the converged solution of Kinematic ICP, which has been verified by visual inspection and by manually measuring the approximate distances in the real world between the scan poses.

4.6. Surface reconstruction

The resulting PCD model after registration contains variable point densities, doubled regions in overlapping areas, and holes resulted due to scanning objects which do not return data (e.g. shiny, black objects). To better illustrate some of these artifacts, we have selected a smaller dataset representing a cup (see Fig. 14).

The surface reconstruction module transforms this PCD model into a homogeneous point distribution suitable for geometrical reasoning, containing interpolated points (which approximate the original input data) that account for the above mentioned problems. To do this we use our Robust Moving Least Squares (RMLS) algorithm [29].

The first step of the algorithm is to compute a set of points which are to be fitted to the surface. If only surface reconstruction is required, the initial guess for the surface can be the original PCD itself. To resample (up- or down-sample) the surface as evenly as possible, a set of equidistant grid points is generated in the proximity of the original PCD, such that holes are covered. As our hole-filling strategy we are using the ideas presented in [40], with the added automation that we fill every hole that is smaller than a given radius. These points will be projected onto the plane fitted through its k nearest neighbors ($p_{1\ldots k}$) from the cloud, thus bringing it in the proximity of the surface. Then, each point q from the projected initial guess Q is fitted to the surface which approximates the original PCD by a bivariate polynomial height function in the local Darboux frame. The frame consists of vectors U, V, N, with $U \perp V \perp N$, N being parallel to the fitted plane's normal.

Because the PCD contains measurement and/or fitting errors, in the first fitting step of RMLS, we apply robust plane fitting using SAmple Consensus methods (see Section 4.2) instead of the usual iterative approaches [1] in order to preserve sharp edges in the cloud [29].

In the second fitting step, for each point $q \in Q$, a set of weights $W_{i=1\ldots k}$ is computed for the k nearest neighbors of q (p_i): $W_i = \exp(-\frac{\|q-p_i\|^2}{h})$, where the h parameter defines how much the surface will be smoothed by influencing the proportion of the weights of close and distant points. If u, v and n are coordinates along the U, V and N, axes of the Darboux frame, we can define a number of x height functions $f^j_{(u,v)}, j = 1\ldots x$ – selected as members of bivariate polynomials – whose sum, when multiplied with the coefficients c_j, approximates the surface in the proximity of q's neighbors (which were weighted by W) such as: $n_{(u,v)} = \sum_{j=1}^{x} c_j \cdot f^j_{(u,v)}$.

To compute the vector of coefficients, we minimize the error function as in [1,40]. Since the reference plane was computed robustly, the height function $n_{(u,v)}$ defined on it preserves the edges sharper than in regular MLS (see Figs. 15 and 16). After resampling, surface normals have to be re-computed as the normal of the fitted polynomial in point q. Usually 2nd order polynomials are good enough to approximate the surface because most of the areas are either planar or curved in only one direction, but when this is not the case, for example when two or more surfaces meet, fitting a higher order polynomial is more appropriate. While it is computationally more expensive to fit 3rd order

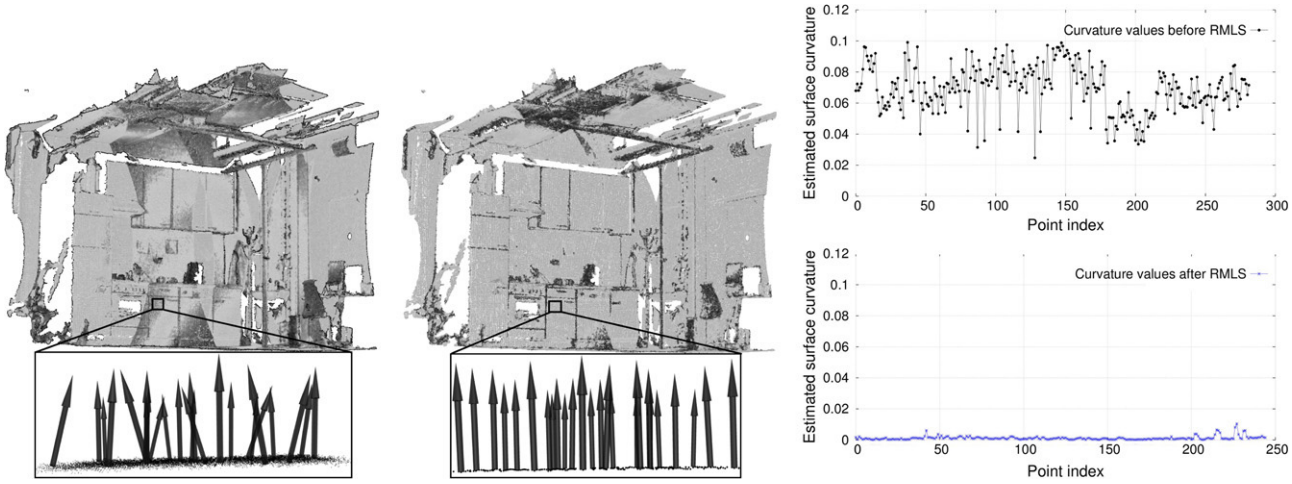

Fig. 16. Resulting surface curvature estimates on: a PCD after registering two partial views (left); the resampled PCD (middle). Notice the errors in the surface normal estimates. The plots on the right depict the estimated surface curvatures before and after resampling. In both scans, curvature values are shown in logarithmic grayscale: low curvatures are lighter while high curvatures are darker.

polynomials, in these occasions the results are better especially if one considers the correctness of the recalculated surface normals. Fitting higher order polynomials is more and more costly as the number of members increases, and the additional precision gained is insignificant.

Fig. 16 presents the resulting surface curvature estimates computed on a PCD model directly after registering two partial views, against the results obtained after resampling the dataset. Note that the quality of the surface curvature is greatly improved, at the small cost of losing some detail in the overlapping region.

5. Functional mapping

The complete PCD world model obtained contains only geometrical information about the world, which does not suffice for performing robot tasks besides navigation and localization. Therefore, the functional mapping modules extract semantic information from the environment map, based on 3D geometry and a set of assumptions about the world. By restricting ourselves to indoor kitchen environments, and by employing a set of probabilistic algorithms, the following general categories of objects need to be extracted: (i) tables — as planar horizontal surfaces located roughly at hip height; (ii) cupboards and drawers — as vertical surfaces with a certain area and having a handle; (iii) kitchen appliances — as cupboard-like surfaces with knobs and handles. All other regions, including chairs, will be considered as obstacles, as they play no important role for our robot. The functional mapping module is comprised of the following steps:

- a segmentation step for extracting regions with certain geometrical characteristics from the input dataset (e.g. planar areas);
- polygonal fitting for representing the extracted regions in a more compact format as well as approximating planar surfaces meeting certain criteria with cuboids;
- a handle and knob identification and extraction step for determining whether the approximated cuboids are candidates for a certain object class;
- functional reasoning for assigning a candidate to an object class using commonsense knowledge (either hardcoded or acquired through learning).

5.1. Point cloud segmentation

Given an unorganized point cloud P, containing 3D point coordinates, and normal and curvature estimates, the problem is to find 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 [27]. 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 criterion based on which a neighbor p_k is said to belong to the same region is: $\text{acos}(\|n_p - n_{p_k}\|) \leq \alpha_{th}$, where n_p and n_{p_k} are the 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 criterion and the queue of seed points is empty (meaning that all points with a small curvature were already considered).

We 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. Because the resampled point cloud used as input data had smooth transitions between points in normal space, various levels of α_{th} did not influence the results significantly. Therefore, we fixed $\alpha_{th} = 5°$, 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. Fig. 17 presents two segmentation results.

Fig. 17. Segmentation results with $\alpha_{th} = 5°$ and $x = 75\%$. Different colors represent different regions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.2. Model fitting

The Model Fitting module approximates the different parts of the kitchen environment using 3D cuboids for containers (cupboards and kitchen appliances), lines for door and device handles, and circles for knobs.

5.2.1. Cuboid fitting

Once a set of planar regions has been identified in the input data, they need to be replaced with polygons that describe the regions using fewer points, thus leading to a more compact representation of the world.

Additionally, from the set of polygons that replace 2D planar surfaces, the cuboid fitting module has to select those which are vertical and horizontal and satisfy a size criterion, and approximate them to 3D rectangular parallelepipeds (cuboids), by projecting the vertical polygons onto the walls (e.g. cupboards). 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. However, since we would like to generalize our methods to other types of planar surfaces in the future, 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 compute its boundary points [2,29]. The results are presented in the left side of Fig. 18. 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 (see Fig. 18 middle).

For particular solutions, such as replacing regions of points with 2D quads, we use our MLESAC estimator for fitting robustly the best 4 lines to the region boundary. 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 assume there exists only one floor and one ceiling plane in the room (i.e. the room does not contain interior staircases or different levels), and take advantage of the fixed Z-axis from the robot to identify the topmost and lowermost horizontal planes as the ceiling and the floor of the room. To identify the walls in the room, we cluster all vertical planar patches (except the cuboid candidates) by their normal direction into two clusters which are perpendicular to each other. The averages of these two clusters will represent two perpendicular directions, the new X and Y axis. Once the XYZ coordinate system is built, we use MLESAC to search for vertical planes which satisfy any of the following conditions: (i) their height covers at least h_{th}; (ii) their area is larger than a_{th}. The vertical planar candidates which are at a maximum distance d_{th} from the nearest parallel wall are projected onto it to form cuboids. In our experiments, we used $h_{th} = 75\% \cdot \text{dist}_{f-c}$, $a_{th} = 1.75$ m^2, and $d_{th} = 1.25$ m, where dist_{f-c} is the room height (i.e. floor-ceiling distance). The results are presented in Fig. 18 (right) and constitute the basis for the environment object mapping.

The presented algorithm works for vertical planar regions (cupboards, drawers), identifying candidates which have to be investigated further (e.g. locating handles and knobs, estimating opening trajectory).

The rest of the remaining surfaces are triangulated using an algorithm similar to [11], with the exception that we don't create an additional voxel structure, but use the raw point cloud data to create the mesh (see Fig. 20 — left).

5.2.2. Extracting handles and knobs

Given a hybrid box-mesh model, 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, but all whose neighbors are inside a planar region (see Fig. 19). We select them for analysis if they are at a distance dist_i from the surface, in between some predefined threshold $d_{\min} \leq \text{dist}_i \leq d_{\max}$, where one would expect a handle.

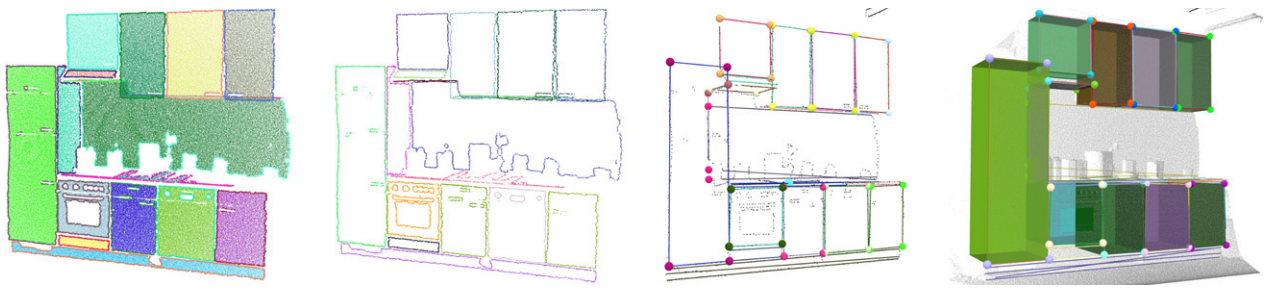

Fig. 18. Boundary points of each segmented planar region (left and center left); 4 best fitted lines to each region boundary (center right); cuboids created by projecting the 2D quad on the wall (right).

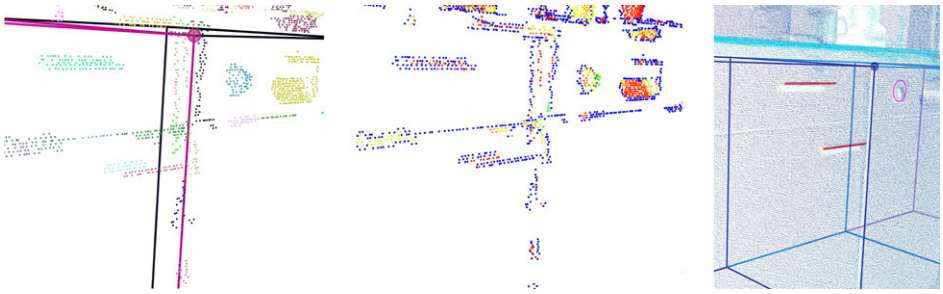

Fig. 19. The remaining clusters of points projected on their associated cuboids (left); Estimated boundary points of each cluster (middle); Handles and knobs segmentation via line and circle fitting (right).

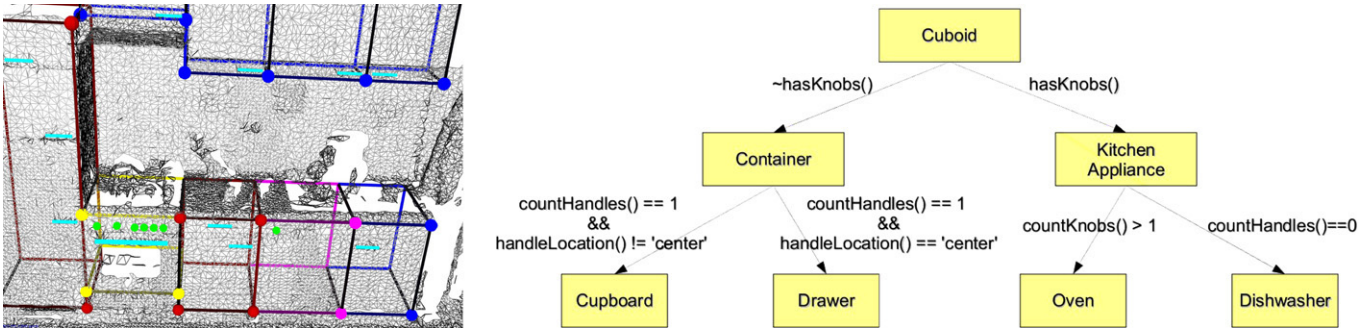

Fig. 20. Left: handles and knobs segmentation for object hypotheses in the kitchen. Right: hierarchical object model.

After projecting the cluster onto the plane, we extract the boundary points and perform a search for a candidate shape. 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. We assume that the projection of a knob will have a circular boundary while the projection of a handle will approximate an elongated shape through which a 2D line can be fit (see Fig. 19 middle). The final model will be selected based on the fitting error produced by each shape (line and circle), and its height from the surface will be dist_i.

As it can be seen in Fig. 19, the scanning quality of the complete PCD model is high enough after registration to locate and 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.

5.3. Functional reasoning

Having a set of geometric primitives (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 resulting hierarchical object model (see Fig. 20) is constructed by defining a set of higher level features which represent the world in a more abstract sense. In our preliminary implementation, we defined the following set of conditions which are tested against each cuboid candidate:

- *hasKnobs()* — true if a cuboid has at least one knob associated with it;
- *hasHandle()* — true if a cuboid has at least one handle associated with it;
- *countKnobs()* — the number of knobs associated with the cuboid;
- *countHandles()* — the number of handles associated with the cuboid;
- *handleLocation()* — 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.

To illustrate how such a feature set can be used to build a hierarchical classification system, we selected a few commonsense rules that gave promissing results in our application scenario. These rules classify a cuboid container with knobs into a kitchen appliance, and without knobs into a simple storage space. A further set of used assumptions describes an oven as a kitchen appliance with one large handle and multiple knobs, and a dishwasher as an appliance with at least one knob and without a handle. Similarly, any container with multiple handles will be considered *undersegmented*, in the sense that multiple handles are uncommon on one door. In this case, the systems triggers that additional sensor information in those areas is required. Fig. 2 represents the hierarchical rule-based classification system used in our application. Saving the acquired information in OWL-DL format makes it easier for us to query the map for answers about the location and number of objects and the relationships between them (see Fig. 21).

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. One approach that we implemented 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. Fig. 22 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

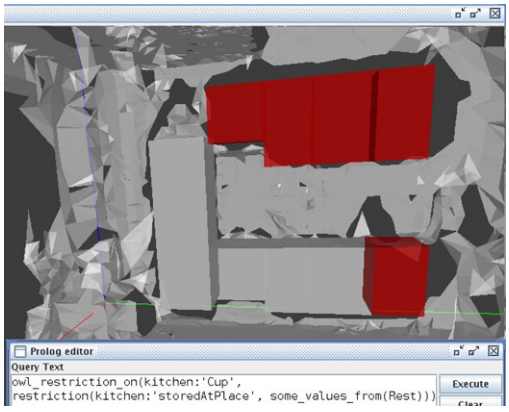

Fig. 21. The results obtained by querying the OWL-DL representation to locate containers where cups can be stored.

to obtain as the scan resolution can be decreased and only smaller regions of interest need to be captured.

The final object model map was automatically imported into a 3D simulation environment (Gazebo) as seen in Fig. 23, by generating the object classes and hinges. We are currently using the simulation engine for task planning and optimization [3].

6. Discussions and overall experimental evaluation

The individual components of the system have been evaluated and empirical results were presented in their respective sections. With regards to time constraints, we view the acquisition of an environment object 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 is 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 that we would have to work with more coarse grained

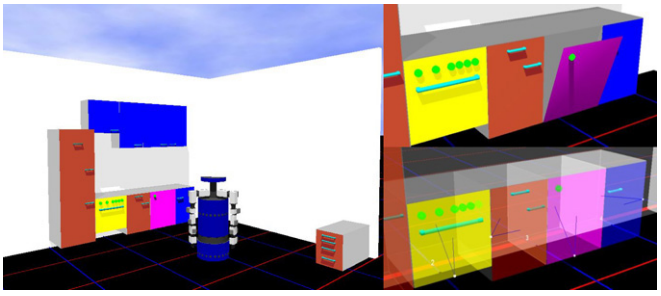

Fig. 23. Functional object map automatically imported in the Gazebo 3D simulator (left) and the identified opening positions and hinges (right).

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 intent to investigate the use of complementary sensors including cameras, as well as active map acquisition. We farther 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.

7. Conclusions and future work

We have presented a system for building environment object maps of indoor household environments (i.e. kitchens). Given partial views of the environment as point clouds, we have developed

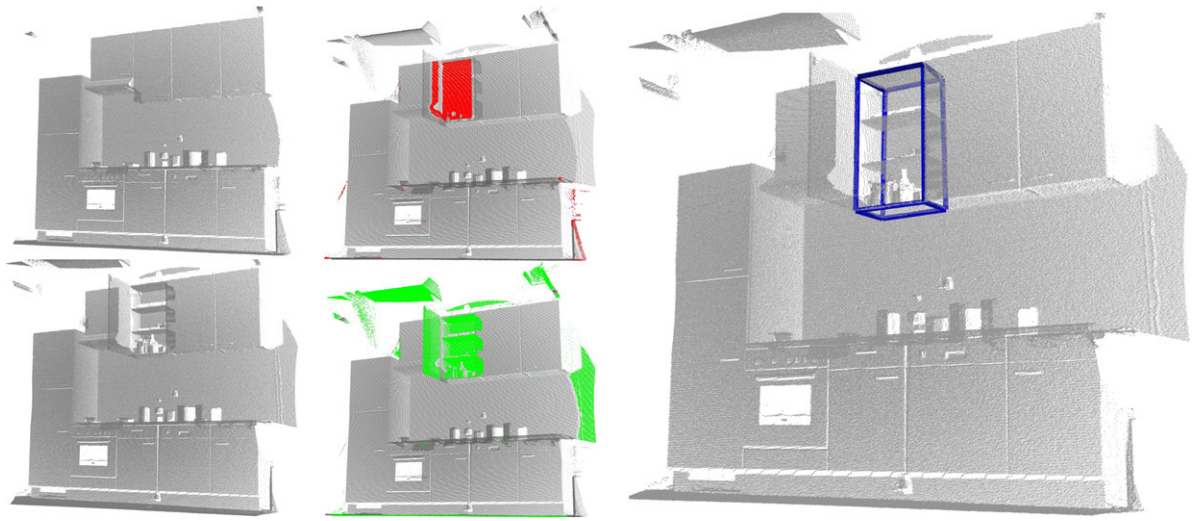

Fig. 22. From top to bottom and left to right: two scans of the kitchen scene depicting two states of a cupboard (closed and open), the highlighted temporal differences in both scans and segmented container.

and extended techniques for statistical analysis, feature extraction, registration, resampling, segmentation, and geometrical and functional reasoning that give promising results for applications such as ours.

One of the main problems that was successfully solved by our methods was the improvement of point cloud data through sparse outlier removal and resampling, while at the same time preserving the information necessary for valid data interpretations, such as the gaps between cupboard doors which are very difficult to distinguish from noise.

Due to the improved point cloud data representations and the interpretation algorithms, we could correctly recognize and localize relevant kitchen objects including cupboards, kitchen appliances, and tables. The cuboid-like objects were classified into different classes based on a set of high-level features: *hasHandle()*, *hasKnobs()*, *countHandles()*, *countKnobs()*, and *handleLocation()*. We are currently investigating techniques for learning object classes conditions automatically, by observing multiple kitchen environments over time.

We consider the development of map acquisition routines that can reliably recognize objects such as cupboards, drawers, etc, as an essential building block for the realization of autonomous robots capable of building semantic maps of their environments. We are currently investigating ways of adding more semantic information to our maps, by making use of sensor networks as we presented in [3,31], 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) [31].

Acknowledgment

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

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C.T. Silva, Computing and rendering point set surfaces, IEEE Transactions on Visualization and Computer Graphics 9 (1) (2003) 3–15.
[2] K.-H. Bae, D.D. Lichti, Automated registration of unorganized point clouds from terrestrial laser scanners, International Archives of Photogrammetry and Remote Sensing (IAPRS) (2004).
[3] M. Beetz, J. Bandouch, A. Kirsch, A. Maldonado, A. Müller, 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.
[4] P.J. Besl, R.C. Jain, Segmentation through variable-order surface fitting, IEEE Transactions on Pattern Analysis and Machine Intelligence 10 (2) (1988) 167–192.
[5] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (2) (1992).
[6] A.W. Fitzgibbon, Robust registration of 2D and 3D point sets, in: Proceedings of the British Machine Vision Conference, 2001.
[7] S. Fleishman, D. Cohen-Or, C.T. Silva, Robust moving least-squares fitting with sharp features, in: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, 2005.
[8] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, et al. Multi-hierarchical semantic maps for mobile robotics, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Edmonton, Canada, 2005.
[9] T. Gatzke, C. Grimm, M. Garland, S. Zelinka, Curvature maps for local shape comparison, in: SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005, SMI' 05, 2005.
[10] N. Gelfand, N.J. Mitra, L.J. Guibas, H. Pottmann, Robust global registration, in: M. Desbrun, H. Pottmann (Eds.), Eurographics Association, ISBN: 3-905673-24-X, 2005.
[11] M. Gopi, S. Krishnan, A fast and efficient projection-based approach for surface reconstruction, in: SIBGRAPI '02: Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing, 2002.
[12] M. Gopi, S. Krishnan, C.T. Silva, Surface reconstruction based on lower dimensional localized delaunay triangulation, in: M. Gross, F. R. A. Hopgood (Eds.), Computer Graphics Forum (Eurographics 2000), vol. 19(3), 2000.
[13] 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.
[14] A. Gruen, D. Akca, Least squares 3D surface and curve matching, International Journal of Photogrammetry and Remote Sensing 59 (2005) 151–174.
[15] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, in: SIGGRAPH '92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, 1992.
[16] A. Johnson, M. Hebert, Using spin images for efficient object recognition in cluttered 3D scenes, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (5) (1999) 433–449.
[17] A.E. Johnson, 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, 1997.
[18] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, 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.
[19] A. Makadia, A.I. Patterson, K. Daniilidis, Fully automatic registration of 3D point clouds, in: CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006.
[20] J. Modayil, B. Kuipers, Bootstrap learning for object discovery, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS-04, 2004.
[21] O.M. Mozos, A. Rottmann, R. Triebel, P. Jensfelt, W. Burgard, Semantic labeling of places using information extracted from laser and vision sensor data, in: Proceedings of the IEEE/RSJ IROS Workshop: From Sensors to Human Spatial Concepts, Beijing, China, 2006.
[22] A. Nüchter, K. Lingemann, J. Hertzberg, Cached k-d tree search for ICP algorithms, in: 3DIM '07: Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling, 3DIM 2007, 2007.
[23] A. Nüchter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, H. Surmann, 3D mapping with semantic knowledge, in: RoboCup, 2005.
[24] A. Nuechter, H. Surmann, J. Hertzberg, Automatic model refinement for 3D reconstruction with mobile robots, in: Proceedings of the 4th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling, 3DIM'03, Banff, Canada, October, 2003.
[25] M. Pauly, M. Gross, L. Kobbelt, Efficient simplification of point-sampled surfaces, in: Proceedings of IEEE Visualization, 2002.
[26] H. Pottmann, S. Leopoldseder, M. Hofer, Registration without ICP, Computer Vision and Image Understanding 95 (1) (2004) 54–71.
[27] T. Rabbani, F. van den Heuvel, G. Vosselmann, Segmentation of point clouds using smoothness constraint, in: International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 36, part 5, Dresden, Germany, September 25–27, 2006.
[28] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, 3-D digital imaging and modeling, 2001, in: Proceedings. Third International Conference on 2001, pp. 145–152.
[29] R.B. Rusu, N. Blodow, Z. Marton, A. Soos, 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, October 29 – 2 November, 2007.
[30] R.B. Rusu, N. Blodow, Z.C. Marton, M. Beetz, Aligning point cloud views using persistent feature histograms, in: Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Nice, France, September 22–26, 2008 (in press).
[31] R.B. Rusu, B. Gerkey, M. Beetz, Robots in the kitchen: Exploiting ubiquitous sensing and actuation, Robotics and Autonomous Systems Journal in: Network Robot Systems (special Issue) (2007).
[32] R.B. Rusu, Z.C. Marton, N. Blodow, M. Beetz, Persistent point feature histograms for 3D point clouds, in: Proceedings of the 10th International Conference on Intelligent Autonomous Systems, IAS-10, Baden-Baden, Germany, 2008.
[33] A. Sappa, M. Devy, Fast range image segmentation by an edge detection strategy, in: Proceedings of Third International Conference on 3-D Digital Imaging and Modeling, 2001.
[34] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, Computer Graphics Forum 26 (2) (2007) 214–226.
[35] G. Sharp, S. Lee, D. Wehe, ICP registration using invariant features, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 24 (1) (2002) 90–102.
[36] P. Torr, A. Zisserman, MLESAC: A new robust estimator with application to estimating image geometry, Computer Vision and Image Understanding 78 (2000) 138–156.
[37] R. Triebel, Óscar Martínez Mozos, 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.
[38] S. Vasudevan, S. Gächter, V. Nguyen, R. Siegwart, Cognitive maps for mobile robots-an object based approach, Robotics and Autonomous Systems 55 (5) (2007) 359–371.

[39] E. Wahl, U. Hillenbrand, G. Hirzinger, Surflet-pair-relation histograms: A statistical 3D-shape representation for rapid classification, in: Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling, 2003.
[40] J. Wang, M.M. Oliveira, A hole-filling strategy for reconstruction of smooth surfaces in range images, in: XVI Brazilian Symposium on Computer Graphics and Image Processing, SIBGRAPI 00, 2003, 11.
[41] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, International Journal of Computer Vision 13 (2) (1994) 119–152.

Radu Bogdan Rusu is a Ph.D. student/researcher at the Intelligent Autonomous Systems group, Computer Science Department IX, Technische Universitaet Muenchen. He received his Masters studies (2004) and Postgraduate studies (2005) degrees from Technical University of Cluj-Napoca, Romania. He is currently affiliated with the SRI Artificial Intelligence Center as an International Visiting Researcher, working in the area of perception and 3d semantic mapping. During the last couple of years, he served as a developer and contributor for the Player project, as well as maintainer of the open source Javaclient library. Radu Bogdan Rusu was one of the main organizers of the first Player Summer School for Cognitive Robotics (PSSCR'07), which brought together over 50 international students from all over the world. He also served as a reviewer for robotics conferences such as IROS or ICRA and is a member of IEEE Robotics and Automation Society.

Zoltan Csaba Marton is a Ph.D. student/researcher at the Intelligent Autonomous Systems Group, Computer Science Department IX, Technische Universitaet Muenchen. He received his Masters degree in 2007 from Technical University of Cluj-Napoca, Romania, with his thesis entitled "Object Reconstruction and Semantic Mapping Using Point-Set Surfaces". Zoltan co-authored several publications in the field of geometrical reasoning from 3d point clouds and is a member of the IEEE Robotics and Automation Society.

Nico Blodow is a Ph.D. student/researcher at the Intelligent Autonomous Systems Group, Computer Science Department IX, Technische Universitaet Muenchen. He received his Diploma degree in Computer Science with distinction in 2008 from Technische Universitaet Muenchen, Germany, on the topic of geometrical reasoning and model fitting in 3D point clouds. His special interest lie in robotics perception and artificial intelligence and he has co-authored several publications in the field of 3d semantic mapping and point cloud processing.

Mihai Dolha is a Ph.D. student/researcher at the Technische Universitaet Muenchen, Computer Science Department IX and is a member of the Intelligent Autonomous Systems group. He he received his Masters studies diploma degree at the Technical University of Cluj-Napoca, Romania in 2007. He is also a member of the IEEE Robotics and Automation Society. His research interests include mobile robots, motion and grasp planning and robot simulators.

Michael Beetz is a professor for Computer Science at the Department of Informatics of the Technische Universitaet Muenchen and heads the Intelligent Autonomous Systems group. He is also managing director of the German cluster of excellence COTESYS (Cognition for Technical Systems) where he is also co-cordinator of the research area "Knowledge and Learning". Michael Beetz received his diploma degree in Computer Science with distinction from the University of Kaiserslautern. He received his M.Sc., MPhil., and Ph.D. degrees from Yale University in 1993, 1994, and 1996 and his Venia Legendi from the University of Bonn in 2000. Michael Beetz was a member of the steering committee of the European network of excellence in AI planning (PLANET) and coordinating the research area "robot planning". He was also principal investigator of a number of research projects in the area of AI-based robot control. His research interests include plan-based control of robotic agents, knowledge processing and representation for robots, integrated robot learning, and cognitive perception.