

## 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.

© 2008 Elsevier B.V. All rights reserved.

### 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 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<sup>1</sup> 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.

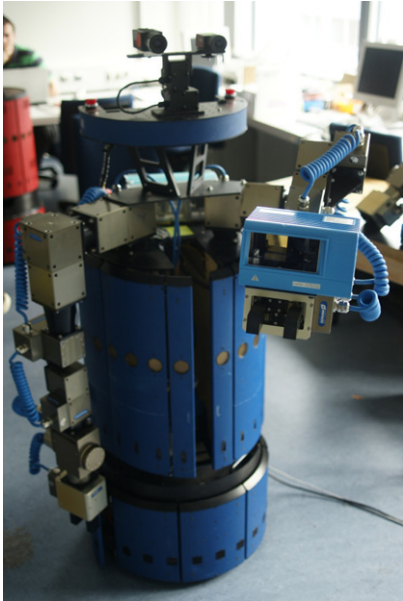
The main challenge of this scenario is the transformation of raw point cloud data into a meaningful, compact representation of the world, without losing fine grained details necessary for tasks like robot manipulation and activity recognition. The key contributions of this article include a novel multi-dimensional 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](mailto:rusu@cs.tum.edu) (R.B. Rusu), [marton@cs.tum.edu](mailto:marton@cs.tum.edu) (Z.C. Marton), [blodow@cs.tum.edu](mailto:blodow@cs.tum.edu) (N. Blodow), [dolha@cs.tum.edu](mailto:dolha@cs.tum.edu) (M. Dolha), [beetz@cs.tum.edu](mailto:beetz@cs.tum.edu) (M. Beetz).

<sup>1</sup> 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.



**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.

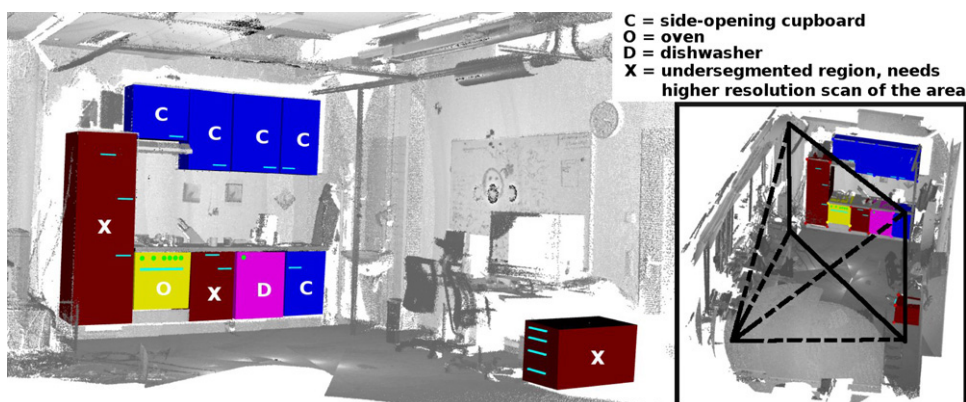
## 2. Related work

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

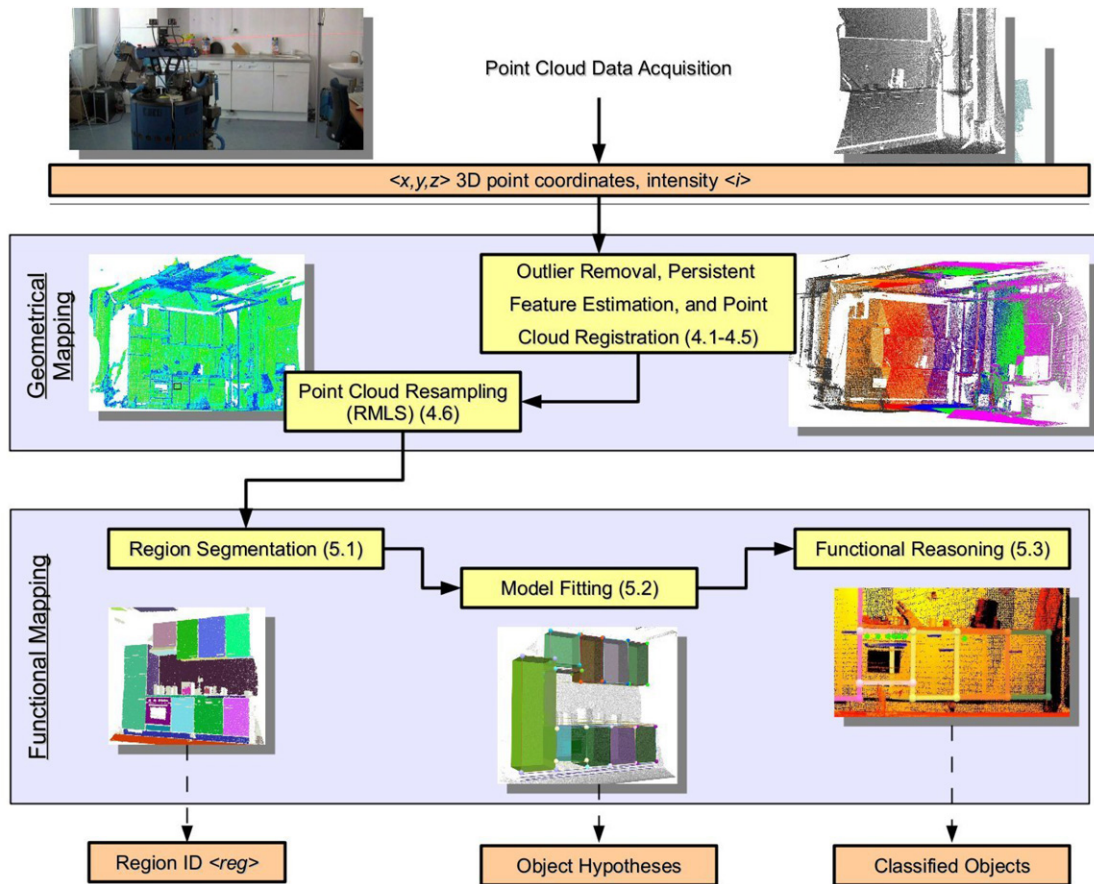


Fig. 3. The computational structure of the acquisition of object maps.

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  $\mu$  and standard deviation  $\sigma$  of nearest neighbor distances, and trimming the points which fall outside the  $\mu \pm \alpha \cdot \sigma$ . The value of  $\alpha$  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 approximately 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  $i = 1 \dots k$ ):

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

followed by the eigenvector  $V$  and eigenvalue  $\lambda$  computation  $C \cdot V = \lambda \cdot V$ . The term  $\xi_i$  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  $\mu$  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

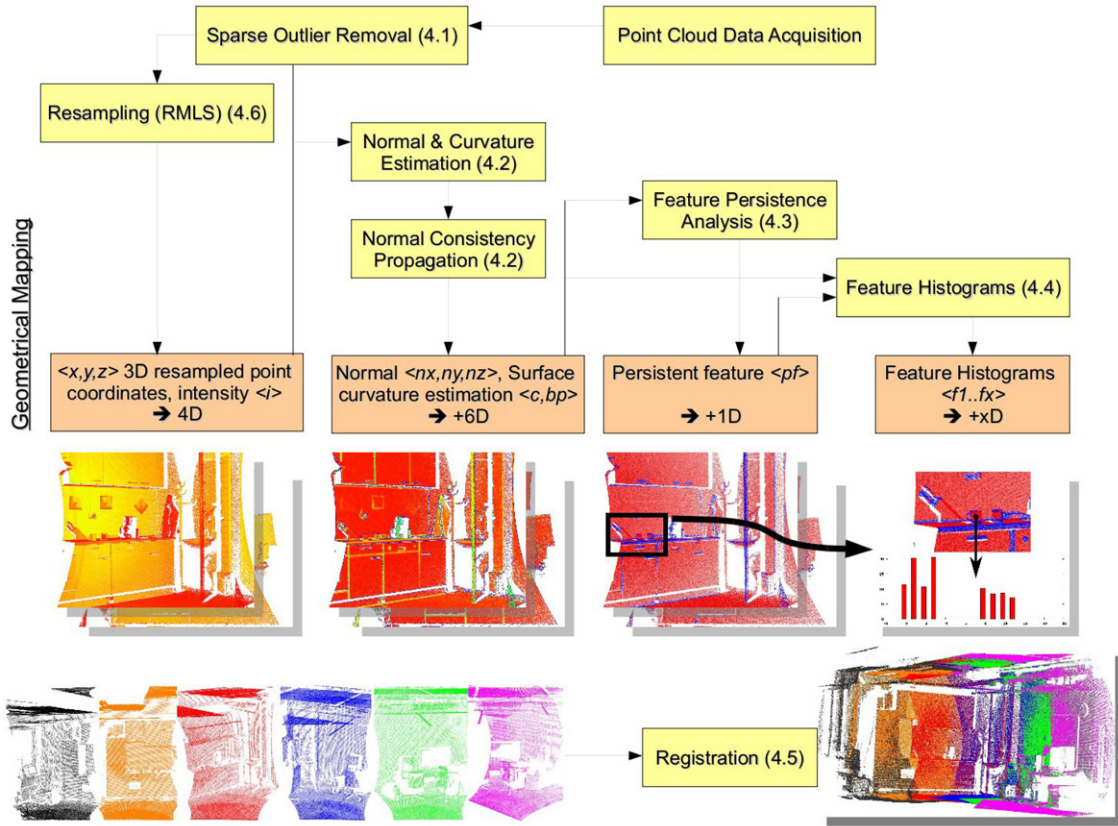


Fig. 4. The architecture of the Geometrical Mapping module.

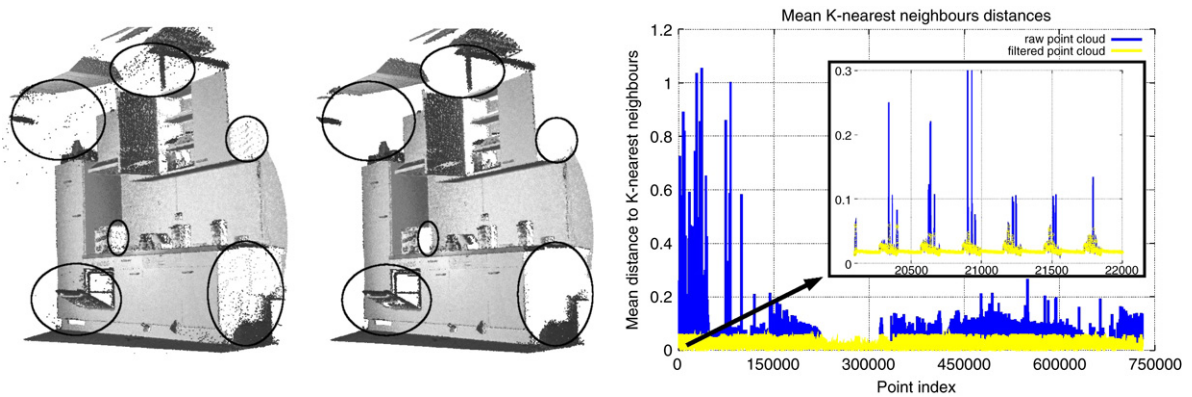


Fig. 5. Left: raw scan; middle: scan after applying gross outlier removal (points left: 728 070 out of 730 664, i.e. 99.64%); right: mean distances to  $k = 30$  neighbors before and after removal using  $\alpha = 1$ . In both scans remission values are shown in grayscale.

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

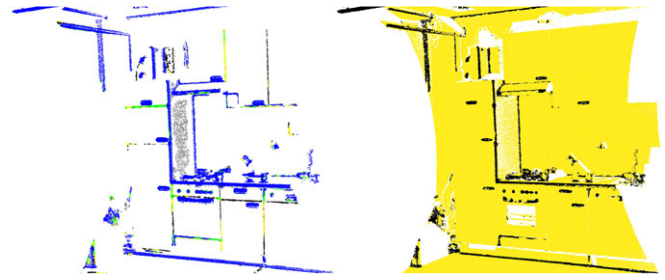
When using point features as characteristics of the entire point cloud, it's good to make sure that we find a compact subset of points that best represents the point cloud. The lesser the number

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 ( $i = 1 \dots m$ ), and compute different values for a feature, say  $\gamma_p$  for each  $r_i$ . We compute the mean  $\mu_i$  and standard deviation  $\sigma_i$  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:

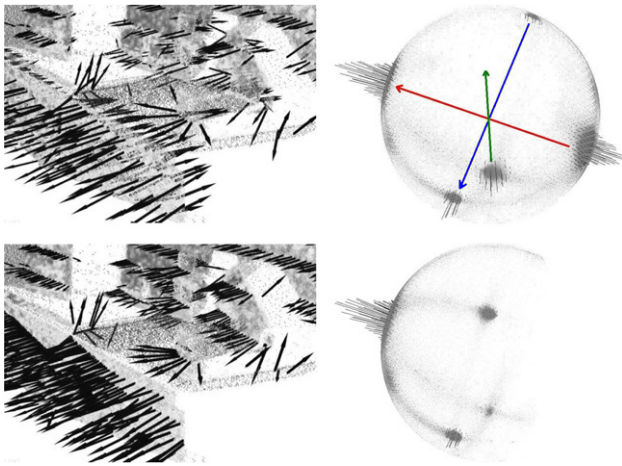
$$P_f = \bigcup_{i=1}^{m-1} (P_{f_i} \cap P_{f_{i+1}}). \quad (2)$$

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. 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.)

**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 EGs: before normal flipping (top row) and after (bottom row).

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.

















