appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

Towards Geometric Mapping for Semi-autonomous Mobile Robots

Georg Arbeiter and Richard Bormann and Jan Fischer and Martin Hägele and Alexander Verl

Institute for Manufacturing Engineering and Automation, Fraunhofer IPA, 70569 Stuttgart, Germany,
georg.arbeiter@ipa.fraunhofer.de

Abstract. Semi-autonomous mobile robots are a promising alternative for tasks that are too challenging for autonomous robots. Especially in an unstructured environment, full autonomy is still far from being realized. In order to enable the human operator to control the robot properly, visualization of the environment is crucial. In this paper, we introduce a pipeline for geometric mapping that uses narrow field of view RGB-D cameras as input source and builds a geometric map of the environment while the robot either is operated manually or moves autonomously. Geometric shapes are extracted from subsequent sensor frames and are clipped and merged in a geometric feature map. Evaluation is done both in simulation and on the real robot.

1 INTRODUCTION

Although performance of fully autonomous robots has improved greatly in recent years, they still fail frequently while solving tasks in unstructured environments. This is because of inaccurate sensors and actuators as well as non-robust algorithms. A promising alternative are semi-autonomous robots that try to fulfill common tasks autonomously until an unexpected situation occurs. In this case, a human operator can compensate for the lack of intelligence and accomplish the task manually. Optimally, the robot is able to learn from the human actions and thus increases its degree of autonomy over time.

For both the human operator and the robot, perception of the environment is inevitable. Whereas the robot needs information about its surroundings for localization, collision avoidance and planning of actions, the human operator needs visualization of both the current field of view of the robot and past sensor data in order to be able to understand the environment. Most of the robot's demands can be met with a point map representation whereas a geometric map is suitable for data transfer over network and visualization. This leads to the need of a hybrid environment model, consisting of a point and a geometric map.

In this paper, we propose a pipeline for geometric mapping of the environment with focus on semi-autonomous robots. Both a point and a geometric representation of the environment are created during processing. As sensors, RGB-D cameras are used. Being very cost efficient and having a high frame rate they are advantageous over previously used tilting laser scanners. However, the narrow field of view demands additional

The Orginal Publication is available at http://link.springer.com/chapter/10.1007/978-3-642-32732-2_7

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

processing steps. The method we propose does most of the calculations on single sensor frames instead of the full map. Hence, we meet the requirements of a continuous data flow and are able to create the map while the robot is moving. This is a special need if a human operator is present as he or she needs immediate feedback of the robot's vicinity. As the movements commanded by the human are unforseeable and may be at a wide range of speed, higher robustness of the mapping is needed than for a fully autonomous mapping.

The first processing step is point cloud registration. The clouds are aligned to the robot's map coordinate system using *Iterative Closest Point* (ICP). This step is followed by iterative extraction of geometric features like planes. Each plane is passed to the geometric map afterwards. We propose a novel method for processing these extracted planes: The surfaces are transformed to a common coordinate system. Afterwards, 2-D polygonal clipping is applied followed by a merging step. The merged planes are adjusted corresponding to their relative pose. This increases the robustness against inaccurate plane extraction. Using a polygonal representation, we offer the chance to provide a clear visualization to the human user. Additionally, user input can be used to correct erroneous maps by selecting single shapes, deleting them or changing their position. This is not possible with a bare point cloud representation.

The key contributions of this paper are (1) a mapping pipeline for single frame processing of RGB-D data in order to create a geometric map, (2) a novel approach for generating a geometric map from extracted planes and (3) a geometric map representation that can easily be understood and modified by a human.

The remainder of this paper is structured as follows: Section 2 provides related work regarding geometric mapping, semi-autonomous behaviour and polygon clipping. in Section 3 we present the mapping architecture and algorithms used. The evaluation of the mapping and results are shown in Section 4. The paper concludes with a resume and an outlook on future work.

2 RELATED WORK

Aggregation of geometric maps from point cloud data has been subject to many research activities in recent time. For example, Rusu et al. used a tilting laser scanner to acquire point clouds in [1] and performed planar segmentation using *Random Sample Consensus* (RANSAC) in order to find table surfaces. Further work by Rusu et al. shows semantic object labeling of planar surface structures in kitchen environments like cupboards, tables and drawers [2]. They also create polygonal represenations of extracted surfaces but do not approach the merging problem related to sequnetial mapping. In [3], Nüchter et al. used a combination of ICP and RANSAC for plane extraction in point clouds in order to create a semantic map. However, they stop at the stage of labeled point representations rather than creating a map consistent of geometric shapes. Another mapping pipeline was proposed by [4]. Henry et al. perform RGB-D SLAM using a Kinect camera. They use feature points from the color image, apply RANSAC and ICP based registration including loop closure and finally create a SURFEL representation without deriving geometric shapes. A method for volumetric mine mapping using occupancy griuds was presented in [5]. All these approaches are targeted at fully

The Orginal Publication is available at http://link.springer.com/chapter/10.1007/978-3-642-32732-2_7

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

autonomous robots and do not propose a map representation suitable for human perception. Some of them create maps in stop-and-go fashion which is not applicable for humanly controlled robots.

Semi-autonomous behaviour in perception is not so extensively researched. Some approaches deal with human interaction in navigation. In [6, 7], virtual objects are augmented in a camera stream and the user can control the robot to avoid obstacles. However, no 3-D mapping is used in order to improve the users immersion. Goodfellow et al. [8] introduced a system that presents the output of the perception module to the user in order to get feedback about the next action. Recently, Pitzer et al. presented an approach for shared autonomy in perception [9]. The user has to identify objects the robot is not able to recognize. Both approaches focus on the field of object detection and have a much stronger user involvement than our approach. Currently, there is no system known to the authors that is able to create geometric map representations from 3-D data in order to satisfy both the requirements of autonomous and tele-operated mode. Basic ideas of our work were already presented in [10], namely point cloud registration and processing of convex hull polygons. This work is extend within this paper.

Processing of polygons is a common task in computer graphics and gaming. A variety of different approaches ([11–13]) to polygon clipping can be found in literature. Comparison of several polygon clipping methods is available in [14]. Most of the methods are limited in the types of polygons they can handle. Also, computational speed varies greatly. A generic solution to 2-D polygon clipping was introduced by Vatti in [15]. His method is able to clip and merge almost any kind of polygons in an efficient way.

3 METHODOLOGY

Our mapping pipeline for geometric maps is designed for the use of narrow field of view RGB-D or time-of-flight cameras. The system processes one sensor frame after another and does not use the full map representations for most of the calculations. Only depth information is currently used. We carefully choose algorithms to achieve on-the-fly processing and keep the computational complexity of the system low since processing of sensor frames during robot movement is essential for a tele-operated robot.

3.1 System Architecture

The system architecture is shown in Fig. 1. The first step is point cloud registration. We use a variant called frustum ICP for alignment. ICP is applied on a downsampled sensor frame to reduce computation time. Also, not every sensor frame is processed but only key frames. The output of the registration component is a point map and an aligned key frame. In the feature extraction step, planes are extracted from each key frame in an iterative way in order to find all planes in the current point cloud. Concave hulls of the planes are passed to the geometric aggregation module. The hull polygons are clipped and merged into the geometric map. Finally, merged polygons are adjusted in pose w.r.t. the input polygons.

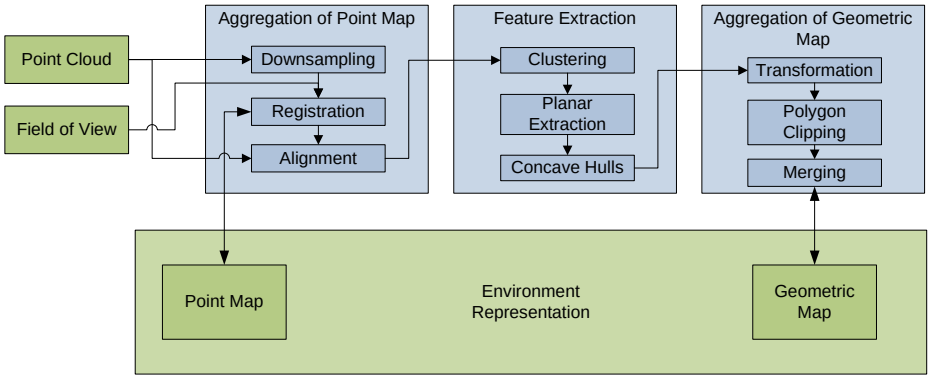


Fig. 1: Architecture of the mapping pipeline.

3.2 Registration

Due to the narrow field of view of the camera sensors and the uncertainty of both camera data and robot position, registration of point clouds is inevitable. We rely on the ICP algorithm which is widely used for registration in robotics. However, to bound computational effort in long-term operation, we use the frustum ICP variant [16] that considers the current field of view of the sensor.

First, we select key frames according to the robot movement. This means that we only allow a new point cloud for registration if the robot moved to a certain extent since the last registration event. Each key frame is downsampled using a voxel filter and aligned to the existing point map. However, not the full map is used but only the part being in the current field of view of the sensor. The field of view is modeled as a frustum as described in [10] and for each point in the map an inside-outside test is performed using the normal vectors of the frustum planes. Once the final transformation is found, the original full resolution point cloud is transformed and passed to the feature extraction. Using key frames, we decrease the chance of mis-alignments that might occur if the robot is moved to fast by a human. Also, we evaluate the fittness score of the registration and reject a frame it is not high enough. In this case, we try aligning the next frame.

3.3 Feature Extraction

The next processing step is feature extraction. As features we consider basic geometric shapes like planes, lines or cylinders. In this paper, we use RANSAC to extract planes from each key frame.

The plane extraction is done in an incremental manner. First, Euclidean clustering is used to determine connected regions of the point cloud. In a second step, we try to fit planes to each cluster using RANSAC. If the plane found has at least a minimum number of inliers, it is considered for further processing, removed from the cluster and

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

the fitting step is repeated. We do this as long as no new plane can be fitted or the cluster size falls below a user defined threshold.

It is also possible to set constraints in order to specify what kind of planes should be extracted. For example, if only horizontal planes are of interest for the geometric map, we only consider planes as valid if their normal vector is approximately parallel to the z axis.

In order to describe the extracted planes we use both the plane coefficients of the cartesian form and a concave hull. First, the inliers of the plane are extracted from the cluster and projected on the plane. For a point p the distance δ to a plane

$$pl : \boldsymbol{n} \cdot (\boldsymbol{x} - \boldsymbol{a}) = 0 \tag{1}$$

is defined as

$$\delta = \boldsymbol{n} \cdot (\boldsymbol{p} - \boldsymbol{a}). \tag{2}$$

The projection of p on the plane follows as

$$\boldsymbol{p_{pr}} = \boldsymbol{p} - \delta \boldsymbol{n}. \tag{3}$$

Second, a concave hull of the projected inliers is constructed using alpha shapes. The hull points are sorted so that it is possible to create a polygon. Afterwards, both the hull polygon and the plane coefficients are passed to the geometric map for further processing.

3.4 Aggregation of geometric map

As the same scene is observed multiple times from different points of view, many of the extracted planes represent the same objects in the environment and therefore overlap. The goal during aggregation of the geometric map is to merge all planes that describe the same plane in the environment. To achieve this, we use 2-D polygon clipping algorithms.

As polygon clipping is a common task in computer graphics, there are many approaches in literature. However, all of them only work for 2-D polygons. Thus, we have to transform polygons into a common coordinate system if we want to clip them. First, we define a similarity measure for the planes in order to determine candidates for merging. For two planes p_1 and p_2 with

$$p_i : a_i x + b_i y + c_i z + d_i = 0,\ i = 1, 2 \tag{4}$$

the normal vector is

$$\boldsymbol{n_i} = \begin{pmatrix} a_i \\ b_i \\ c_i \end{pmatrix} \tag{5}$$

The similarity measure is defined as

$$\|\boldsymbol{n_1} \cdot \boldsymbol{n_2}\| > t_1 \tag{6a}$$

$$d_1 - d_2 < t_2 \tag{6b}$$

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

if the normal vectors of the two planes point in the same direction. If not, one of the normal vectors has to be flipped. The conditions (6) limit the maximum angle and distance deviation. The parameters t_1 and t_2 are user defined and depend on the desired granularity of merging. If the two planes meet the similarity condition, they are transformed into a common coordinate system. We now consider to be p_1 the plane already residing in the map, whereas p_2 is a new plane coming from extraction. The weight δ is introduced for each plane in the map. It is increased after every successful merge to account for the fact that the confidence in this feature increases over time.

The clipping coordinate system of p_1 and p_2 is defined as follows: As the coefficients of the merge candidates are similar but not identical we define a virtual average plane

$$p_3 : \frac{(\delta a_1 + a_2)x + (\delta b_1 + b_2)y + (\delta c_1 + c_2)z + (\delta d_1 + d_2)}{\sqrt{(\delta a_1 + a_2)^2 + (\delta b_1 + b_2)^2 + (\delta c_1 + c_2)^2}} = 0 \tag{7}$$

The weighting factor δ yields a stronger influence of planes that have been merged multiple times before. Thus, the robustness against outlier planes is increased, e.g. if registration is not accurate or plane extraction fails.

The coordinate frame for this plane is defined so that $\boldsymbol{n_3}$ from (5) represents the z-axis $\boldsymbol{z_3}$. The x-axis $\boldsymbol{x_3}$ and y-axis $\boldsymbol{y_3}$ can be chosen freely, they only have to be located on $\boldsymbol{p_3}$ and form a right-hand coordinate system with $\boldsymbol{z_3}$. The origin of coordinate system is calculated by

$$x_o = y_o = z_o = \frac{-(d_1 + d_2)}{n_{3,x} + n_{3,y} + n_{3,z}} \tag{8}$$

which solves the plane equation. The transformation from the world coordinate frame to the p_3 frame can be derived in two steps. First, the rotation matrix is set to

$$\boldsymbol{R} = \begin{pmatrix} x_{3,x} & x_{3,y} & x_{3,z} \\ y_{3,x} & y_{3,y} & y_{3,z} \\ z_{3,x} & z_{3,y} & z_{3,z} \end{pmatrix} \tag{9}$$

using the axes of the p_3 frame. Second, the translation is found by

$$\boldsymbol{t} = \boldsymbol{R} \begin{pmatrix} x_o \\ y_o \\ z_o \end{pmatrix}. \tag{10}$$

The full transformation follows from (9) and (10) as

$$\boldsymbol{T_{w2p}} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \tag{11}$$

Now, all the points $\boldsymbol{pt_w}$ of p_1 and p_2 can be transformed to the common p_3 coordinate frame by

$$\boldsymbol{pt_p} = \boldsymbol{T_{w2p}} \boldsymbol{pt_w} \tag{12}$$

The Orginal Publication is available at http://link.springer.com/chapter/10.1007/978-3-642-32732-2_7

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

For the 2-D polygon clipping, we are only interested in the x and y coordinates of the points. We therefore project the transformed points on p_3 using (3). The projection error is usually small since we only try to merge similar planes and can be ignored.

For clipping, we use the algorithm by Vatti [15]. This method is capable of clipping convex and concave polygons. Also, polygons with holes or self-intersecting polygons can be processed. It uses a generic approach to clip polygons of all kinds: The polygons are parsed in a scan line fashion. While doing this, the edges of the two polygons are marked as left or right bound of the new polygon and also as contributing or not contributing to the new polygon, depending on the occurrence of local minima or maxima. If a left and right bound intersection occurs, edge classification schemes are used to construct the merged polygon.

The algorithm can create both the union or the intersection of polygons. We use this to first check, whether two candidates intersect. If this is the case, the union of the two polygons is calculated and returned as the merged polygon. If the polygons do not intersect, merging is rejected. After merging, the points of the merged plane are finally transformed to the world coordinate system using ${\boldsymbol{T_{w2p}}}^{-1}$ and p_1 is replaced by p_3 in the map. Use of the virtual plane p_3 leads to an adjustment of the plane pose in the map over time. Hence, the pose of inaccurate planes can be improved over multiple merging steps.

4 EVALUATION

In order to proof the functionality of our mapping concept, we do a performance evaluation both on simulated and real data. As test environment we choose the kitchen in our robot lab. The focus of the evaluation is on the geometric mapping part. The registration was already evaluated in [10].

4.1 Setup

As robot, Care-O-bot® 3 (Fig. 3a) is used [17] . It is equipped with a Kinect RGB-D camera on an agile head. Laser range finder based localization is used to provide an estimate for the robot pose. The robot is moved back and forth in front of the kitchen. Both the base and the neck are used to move the camera manually by a human user. This means, the motions of the robot are not planned in advance.

From the simulation, we obtained two datasets, the first of them was recorded while moving the robot in front of the empty kitchen. For the generation of the simulated data, we used the gazebo[1] simulator. It performs ray casting in a virtual environment to simulate a 3-D camera. In order to evaluate robustness, we added Gaussian noise at different magnitude. Set sim_{n0} is without noise whereas sim_{n005} and sim_{n02} have added noise with a standard deviation of 0.005m and 0.02m respectively. In the second dataset sim_{n0obj} we added non-planar objects to the scene in order to make plane extraction more challenging.

[1] http://playerstage.sourceforge.net/gazebo/gazebo.html

The Orginal Publication is available at http://link.springer.com/chapter/10.1007/978-3-642-32732-2_7

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

With the real robot, two datasets were recorded, one similar to the simulated scene without objects $real_1$ (see Fig. 2a) and one with two additional tables in front of the kitchen and non-planar objects added (see Fig. 2b), $real_2$.

(a) (b)

Fig. 2: Kitchen evaluation environment: (a) Set $real_1$, (b) Set $real_2$.

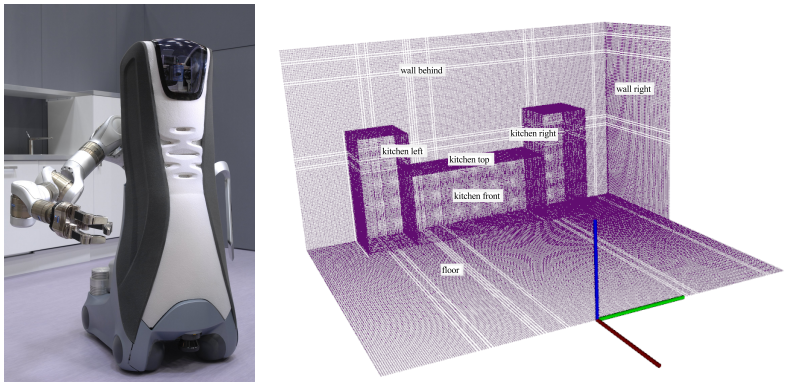
We evaluate the accuracy of the resulting geometric map and the level of plane reduction by merging. Also, we take a look at the mis-detection of planar surfaces if curved surfaces are present. For accuracy evaluation, we created a reference point map of the environment, based on manually measured data, see Fig. 3b. Seven ground truth planes were labeled by hand. The extracted planes are associated with and compared to the ground truth planes. We evaluate the deviation of the plane coefficients d_{coeff}, the angle and distance error d_{angle} and d_{dist} based on the coefficients and the point-to-plane errors RMS_p of the hull points between associated planes. The parameters in (6) are set to $t_1 = 0.95$ and $t_2 = 0.1$.

4.2 Results

The simulated data is used to proof the concept of our mapping algorithm and to test the robustness against noise. The results shown in table 1 can be interpreted as follows: The coefficient, angle and distance deviation grows with increasing noise. This was expected as the planes cannot be fitted as accurate with higher noise than with lower. Also, the point map quality suffers from a higher noise level. However, the point RMS error does not grow as much as one could expect. This is because planes are only merged if they are similar enough. With increasing noise, the deviation between the planes gets higher what leads to less merged planes but also less point errors. The comparison between the empty scene and the one with objects shows that the non-planar surfaces do not disturb the plane extraction and therefore do not influence map quality. On the whole, the values proof a good map at all levels of noise.

The Orginal Publication is available at http://link.springer.com/chapter/10.1007/978-3-642-32732-2_7

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

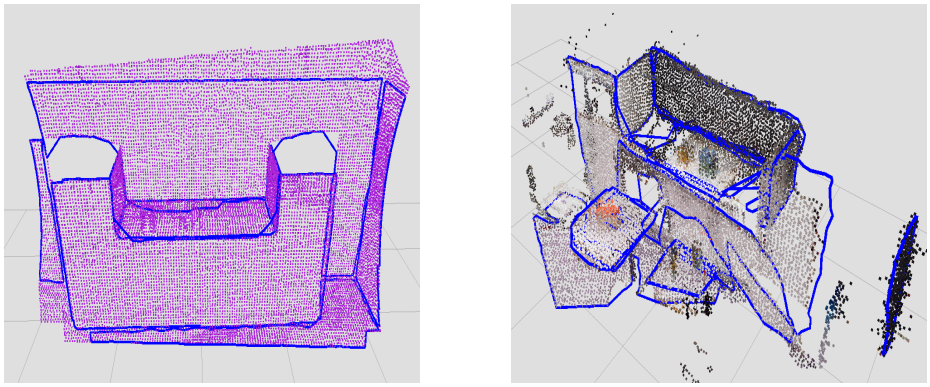


(a) Care-O-Bot® 3 (b) Ground truth

Fig. 3: Mobile service robot Care-O-Bot® 3 (a) and ground truth of the kitchen used for evaluation (b).

Table 1: Accuracy of geometric map for simulated data

set	d_{coeff}	d_{angle} (rad)	d_{dist} (m)	RMS_p (m)
sim_{n0}	0.0108	0.0067	0.0071	0.0054
sim_{n005}	0.0211	0.0057	0.0198	0.0246
sim_{n02}	0.1511	0.0580	0.1368	0.0211
sim_{n0obj}	0.0108	0.0056	0.0084	0.0048



(a) (b)

Fig. 4: Point and geometric map from (a) simulation and (b) real data. Hull polygons of planes marked blue.

The Orginal Publication is available at http://link.springer.com/chapter/10.1007/978-3-642-32732-2_7

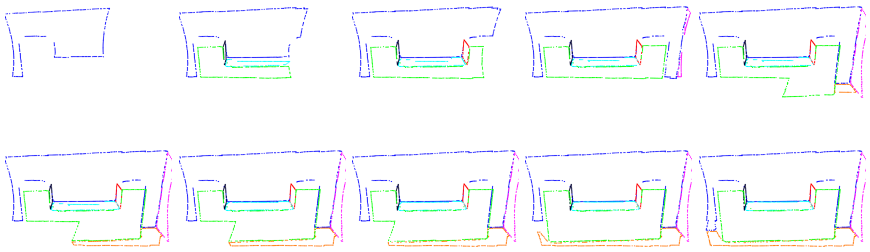


Fig. 5: Merging sequence.

Fig. 4a shows the map for the case without noise. It can be seen that all planes are extracted correctly, that all planes belonging together are merged correctly and that the bounds of the planes correspond well with those of the point cloud. In Fig. 5 the merging sequence is shown. While the robot moves, the geometric map grows and so do the planes when additional parts come into view.

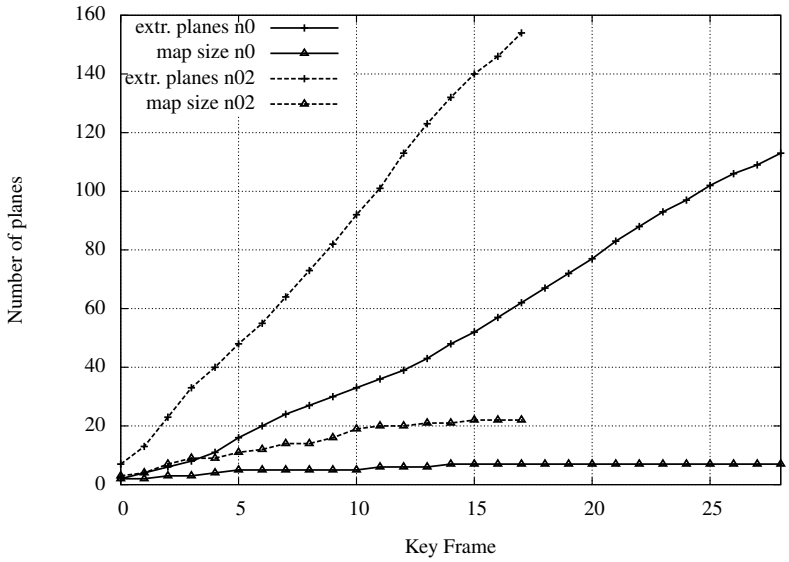


Fig. 6: Map size compared with number of extracted planes.

Another interesting measure is the number of planes in the map compared with the number of incoming planes as it shows the power of the algorithm to reduce the number of multiply observed planes. Fig. 6 shows the number of planes over the number of

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

key frames for the empty kitchen at 0 and 0.02 noise. It can be seen, that the number of planes in the map increases until all planes in the scene have been seen once and then stays constant. The plane reduction ratio is 16.14. At a higher magnitude of noise, more planes are extracted and cannot be reduced to the minimum number of planes. Nevertheless, the plane reduction ratio is still at 7.0. For the real robot, the empty kitchen scene and one with added tables and objects are evaluated. In table 2 can be seen that the deviations of the real scene are at a similar magnitude than those from the simulated scene at a noise level of sim_n02. The performance of the mapping is similar in both scenes, also in the one with additional tables and objects. Hence we can conclude that our algorithm works robustly even in cluttered environments.

Fig. 4b shows the map generated from set $real_2$. It can be seen that most of the planes are placed well. Also, the objects on the tables are not detected as planes. However, merging of the planes is not as perfect as with the simulated data and there is an observable deviation of the kitchen front plane. We will investigate this issue in the following.

Table 2: Accuracy of geometric map for real data

set	d_{coeff}	d_{angle} (rad)	d_{dist} (m)	RMS_p (m)
$real_1$	0.1699	0.0692	0.1020	0.0421
$real_2$	0.1520	0.0658	0.1019	0.0317

The noise of the Kinect camera is comparable to the simulated scene but additionally, the Kinect shows distortion especially in regions that are further away or close to the border of the point cloud. This leads to decreased point map accuracy. The geometric map can never be more accurate than the point map as it uses aligned key frames as input. Thus, we take a closer look on the set $real_2$. Table 3 shows the RMS error per plane for the data set. The kitchen front has a significantly higher RMS error compared to the other planes. The explanation is that there is a relatively high distortion of the point cloud data as the plane is rather large. The other point is that the wall behind the kitchen is dominant when it comes to registration. Because of inaccurate range data, a good registration to the wall yields a poor registration of the kitchen front.

Timings were measured on simulated data while performing a 360° scan of the kitchen environment. The total sequence has a duration of 90 s. The evaluation was run on a PC with Intel Core i7 @2.80 GHz and 6 GB of RAM. The timings were obtained

The Orginal Publication is available at http://link.springer.com/chapter/10.1007/978-3-642-32732-2_7

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

Table 3: RMS point error per plane for data set $real_1$

set	wall behind	floor	kit front	kit top	kit left	kit right
$real_1$	0.0160	0.006	0.0783	0.0149	0.0191	0.0055

in 100 runs, each run registering 52 key frames and extracting 152 planes followed by merging. Table 4 shows the results for each processing step averaged per key frame. It shows, that plane extraction is the most demanding step, whereas the merging needs almost no time. The procesing time for all steps is 0.194 s enabling the mapping system to run at approximately 5 Hz. As only key frames are processed, the system can run at full Kinect frame rate and fast robot movement.

Table 4: Computation time in s of the mapping per key frame.

Registration	Plane extraction	Merging	All
0.027	0.165	0.002	0.194

Finally, we take a look on the data reduction potential using the proposed method. For the dataset sim_{n02}, 17 key frames are registered which results in a total of 5222400 raw points. Having three 32 bit values per point, the complete size of the data handled is 62.7 MB. Downsampling after registration reduces the size of the point map to 49572 points or 595 kB. Eventually, the resulting geometry map consists of 535 Points plus 4 parameter values for each polygon. Thies yields a total size of 6.5 kB. The huge potential in data reduction becomes clear if we take a look on the relative numbers: From the raw data points to geometric represenatation, the amount of data is reduced to 0.01% of the original size. Table 5 shows the absolute and relative numbers for data reduction.

5 CONCLUSION

We presented a novel pipeline for geometric mapping with RGB-D data as input. The processing includes point cloud registration, extraction of planar surfaces, construction

The Orginal Publication is available at http://link.springer.com/chapter/10.1007/978-3-642-32732-2_7

appeared in Stachniss, Cyrill (Ed.); u.a.: Spatial cognition VIII. LNAI 7463, S. 114-127. Berlin;Heidelberg: Springer, 2012

Table 5: Data reduction

representation	raw data	point map	geometric map
number of points	5,222,400	49,572	535
amount of data (bytes)	62.7 MB	595 kB	6.5 kB
reduction to (%)	100	0.95	0.01

of concave hulls and aggregation of a geometric map. A 2-D polygon clipping algorithm is used to merge new features to the map.

We evaluated the performance of our mapping both in simulation and on real sensor data and showed that all relevant planes in the scenes were detected with sufficient accuracy. We also showed that the mapping is robust against noise.

For the future, several extensions and improvements are possible. For example, updating the map in a dynamic scene is an interesting and important topic. The current field of view could be used to replace both parts of the point and the geometric map if the environment changes. Another extension would be to use additional geometric shapes like lines or cylinders in order to describe non-planar parts of the environment. To achieve this, concurrent extraction of multiple feature types and an extended geometric map have to be created.

6 ACKNOWLEDGEMENTS

This research was partly financed by the EU FP7-ICT-247772 Multi-Role Shadow Robotic System for Independent Living and by the research program ”Effiziente Produktion durch IKT” of the Baden-Württemberg Stiftung, project ”ATLAS”.

References

1. R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments,” in *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, p. 1–6, 2009.
2. R. B. Rusu, Z. C. Marton, N. Blodow, A. Holzbach, and M. Beetz, “Model-based and learned semantic object labeling in 3D point cloud maps of kitchen environments,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, MO, USA*, 2009.

3. A. Nüchter, H. Surmann, and J. Hertzberg, "Automatic model refinement for 3D reconstruction with mobile robots," in *3DIM 2003*, pp. 394–401, Los Alamitos, Calif.: IEEE Computer Society, 2003.
4. P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *the 12th International Symposium on Experimental Robotics (ISER)*, 2010.
5. S. Thrun, D. Hahnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker, "A system for volumetric robotic mapping of abandoned mines," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 3, p. 4270–4275, 2003.
6. J. Carff, M. Johnson, E. El-Sheikh, and J. Pratt, "Human-robot team navigation in visually complex environments," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, p. 3043–3050, 2009.
7. A. Cherubini, G. Oriolo, F. Macri, F. Aloise, F. Babiloni, F. Cincotti, and D. Mattia, "Development of a multimode navigation system for an assistive robotics project," pp. 2336–2342, IEEE, Apr. 2007.
8. I. Goodfellow, N. Koenig, M. Muja, C. Pantofaru, A. Sorokin, and L. Takayama, "Help me help you: interfaces for personal robots," in *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction*, p. 187–188, 2010.
9. B. Pitzer, M. Styer, C. Bersch, C. DuHadway, and J. Becker, "Towards perceptual shared autonomy for robotic mobile manipulation," in *ICRA*, pp. 6245–6251, 2011.
10. G. Arbeiter, M. Hägele, and A. Verl, "Incremental field of view dependent registration of point clouds and extraction of table-tops," in *2011 IEEE/ICRA International Conference on Robotics and Automation, 2011.(ICRA 2011). Workshop on Semantic Perception, Mapping and Exploration*, 2011.
11. K. Holwerda, "Complete boolean description." http://boolean.klaasholwerda.nl/algdoc/top.html, 1998.
12. K. Schutte, *Knowledge Based Recognition of Man-Made Objects*. PhD thesis, University of Twente, 1994.
13. B. Zalik, M. Gombosi, and D. Podgorelec, "A quick intersection algorithm for arbitrary polygons," in *SCCG98 Conf. on Comput. Graphics and it's Applicat.*, pp. 195–204, 1998.
14. M. Leonov, "PolyBoolean . comparison...." http://www.complex-a5.ru/polyboolean/comp.html, 1998.
15. B. R. Vatti, "A generic solution to polygon clipping," *Commun. ACM*, vol. 35, p. 56–63, July 1992.
16. J. Luck, C. Little, and W. Hoff, "Registration of range data using a hybrid simulated annealing and iterative closest point algorithm," in *IEEE International Conference on Robotics and Automation*, vol. 4, p. 3739–3744, 2000.
17. C. Parlitz, M. Hägele, P. Klein, J. Seifert, and K. Dautenhahn, "Care-o-bot 3 - rationale for human-robot interaction design," in *Proceedings of 39th International Symposium on Robotics (ISR), Seoul, Korea*, 2008.