Automatic Construction of Polygonal Maps From Point Cloud Data

Thomas Wiemann, Andres Nüchter, Kai Lingemann, Stefan Stiene, and Joachim Hertzberg

***Abstract*— This paper presents a novel approach to create polygonal maps from 3D point cloud data. The gained map is augmented with an interpretation of the scene. Our procedure produces accurate maps of indoor environments fast and reliably. These maps are successfully used by different robots with varying sensor configurations for reliable self localization.**

I. Introduction

Robotic maps are the basis for all actions of a mobile robot. They are needed for the integral purposes of self localization and path planning. Since manual environment mapping is a tedious job, the robotic mapping problem has drawn a lot of attention in the research community. Recently, the focus shifted from planar 2D maps towards 3D mapping. 3D maps outperform 2D maps for many purposes, such as obstacle avoidance, object recognition and scene understanding.

The rapid development of mobile 3D laser scanners provided the basis for mapping large areas accurately. The resulting 3D maps are point clouds, sampling the surfaces of the environments. Although the sampling density of modern laser scanners increases, the point clouds do not represent continuous surfaces. Related to that, the amount of collected data becomes difficult to handle. Modern scanners produce several millions of data points per scan. One approach to compress the information obtained from laser scanners is to represent the scanned surfaces by means of mathematical descriptions or primitive shapes like triangles or quads.

Most commonly, surfaces are approximated by polygonal meshes, particularly triangle meshes, a standard data structure in computer graphics to represent 3D objects. In this community, various automatic mesh generation procedures have been developed. A wide variety of applications apply these algorithms, e.g., model generation for video games or movies, the accurate documentation of architectural heritage and reverse engineering. These algorithms generate highly accurate polygonal models whose appearance has to be as close as possible to the original object, requiring a lot of computation power. In robotics, however, computing time is critical. Furthermore, high level of detail is not needed in many robotic algorithms, such as localization.

This paper presents an approach to generate polygonal environment maps that can be used for localization as well as for visual inspection of the scanned environments. We demonstrate the usability of these maps in a rescue-like scenario: A small robot equipped with a 3D laser scanner takes several scans of an unknown environment. The acquired scans are registered into a global coordinate system by a 6D SLAM algorithm. Based on the resulting point cloud, a polygonal map is computed using a modified Marching Cubes algorithm [9]. This map is used by another, larger robot to localize itself in the mapped environment.

Our solution focuses on reducing the computational costs and exploits the inherent structure of scenes scanned by a mobile robot: Commonly, robotic mapping operates in environments with mainly planar surfaces. This planarity constraint is utilized by the Marching Cubes algorithm. The resulting map consists of planar polygons that are labeled as walls, floors and ceilings. The geometrical information can be used for localization using ray tracing techniques. Besides these algorithmic advantages, the surfaces can be rendered with standard textures according to their classification to deliver the operator a more realistic impression of the explored area than the original point cloud.

In the remaining text, section II presents previous and related work, section III describes the map generation and labeling algorithm. Sections IV and V present experimental results and an application example. Section VI concludes.

Corresponding author: Thomas Wiemann, Knowledge-Based Systems Research Group of the Institute of Computer Science, University of Osnabrück, Germany. twiemann@informatik.uni-osnabrueck.de

II. Related and Previous Work

A. Robotic Mapping

Mapping algorithms differ in the type of maps used. State of the art for metric maps are probabilistic methods that use two dimensional grid maps, where the robot has probabilistic motion and perception models [15]. Localization then works by integrating these two distributions with a Bayes filter, e.g., Kalman or particle filters. Closed loops, i.e., a second encounter of a previously visited area in the environment, play a special role in mapping. Once detected, they enable the algorithms to bound the error by deforming the already mapped area such that a topologically consistent model is created.

Building 3D maps by means of 3D laser scanners requires to have some version of geometrically consistent 3D point cloud of the environment. In our work, we use a 6D SLAM method and software, described, e.g., in [13]. The 6D SLAM takes 3D scans of the complete environment and registers them into a globally consistent and correct 3D map. Registration has to compensate the fact that every single scan pose is given in 6 degrees of freedom (DoF), i.e., registration has to consider three translation and three rotation dimensions. Our SLAM method uses an optimized ICP implementation for online registration based on odometry estimation. Closed

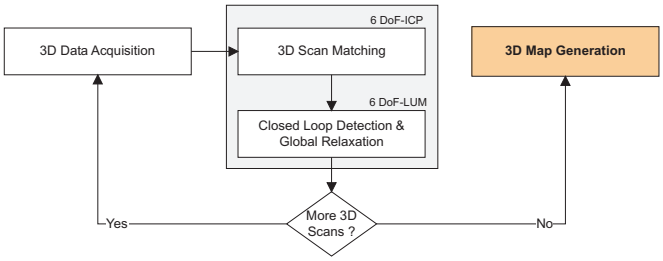

Fig. 1. Mapping system overview. The input of our algorithm is a globally consistent 3D point cloud acquired by a mobile robot solving SLAM.

loops in the given trajectory are detected in an offline post-processing step that distributes the pose differences using a 3D version of Lu and Milios' technique [3], [10].

B. Surface Reconstruction

For surface reconstruction from point cloud data, two main approaches are in use: Detection of primitive shapes or mesh based approaches. Most shape based approaches use feature descriptors to detect instances of predefined objects, e.g., [6]. Other approaches use Hough transformations to detect planar surfaces in 3D laser scans [2].

Mesh based approaches create triangle meshes to approximate the scanned surfaces. The de-facto standard is the Marching Cubes method introduced by Lorensen et al. [9]. This algorithm sub-divides the scanned volume into cubic cells. For each cell the intersections between the cell edges and the surface are calculated. Pre-calculated surface patterns are then used to generate a local triangle mesh approximation. To interpolate the intersections, implicit continuous surface representations like planes or splines are fitted to the local data using least squares fits [1], [8].

A feature of the Marching Cubes algorithm is that it produces far more triangles than are needed to represent an object. Hence, several mesh simplification algorithms have been introduced over the past years. Most of them define error metrics that indicate the error that a certain operation causes to the model, i.e., the removal of an edge [5], [11]. To optimize the model, the edges causing minimal error to the topology are removed iteratively. Since after each edge removal new vertices have to be inserted into the mesh, the initial topology can be altered.

Mesh based surface representations are flexible and able to approximate arbitrary surfaces since they are not limited to predefined object classes. In the following section we will present a fast and reliable mesh based map generation procedure that is based on Marching Cubes and Hoppe's interpolation method.

III. The Polygonalization Procedure

Fig. 1 shows the basic architecture of our mapping system. The single laser scans are registered using ICP and loop closing techniques. After the whole scene is scanned, the output of the 6D SLAM process (i.e. a consistent point cloud) is post-processed to a polygonal map.

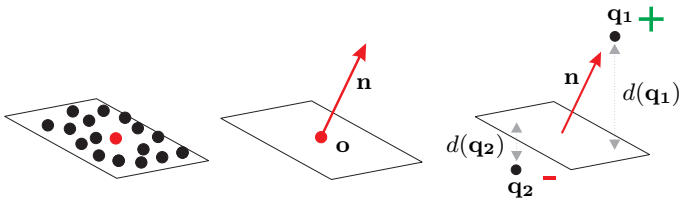

Fig. 2. The construction of the signed distance function as described by Hoppe. For each data point, a so called "tangent plane" is calculated by a least square fit to its k nearest points (left). Each plane is defined by its centroid $\mathbf{o}$ and surface normal $\mathbf{n}$ (middle). The signed distance of a query point is the distance of its projection onto the nearest tangent plane $d(\mathbf{q_i})$ and itself. The sign depends on which side of the surface the query point is (right).

The polygonalization procedure itself consists of three different steps: First, a triangle mesh, based on a consistent 3D point cloud, is generated using the Marching Cubes method. This initial mesh is then optimized by detecting connected planar surfaces. The triangles of such areas are fused to polygonal shapes. This way a compact 3D polygonal representation of the scanned environment is created that is suitable for robotic purposes like localization. After simplification the extracted polygons are finally semantically classified into the categories "floor", "ceiling" and "wall". These three steps detailed in the following sections.

A. Initial Mesh Generation

In the first step, we generate a triangle mesh approximation of the scanned surfaces using the Marching Cubes method and Hoppe's distance function [8]. The idea of Hoppe's approach is to assign a tangent plane $T(p_i)$ to each data point using a local least squares fit to the k nearest points (*k-neighborhood*). These fit planes are represented by the center of gravity o_i of the k-neighborhood and the surface normal n_i:

$$T(\mathbf{p}_i) = \mathbf{o}_i \cdot \mathbf{n}_i.$$

The signed distance of any spatial point $\mathbf{p}$ is defined as

$$d_T(\mathbf{p}) = s(\mathbf{p}) \cdot d(\mathbf{p}, T),$$

where $d(\mathbf{p}, T)$ is the distance of this point to the nearest tangent plane

$$d = (\mathbf{p} - \mathbf{o}_i) \cdot \mathbf{n}_i$$

and $s(\mathbf{p})$ is the sign of the signed distance function according to the relative position of $\mathbf{p}$. This sign is determined by using the orientation of the normal of the tangent plane. If $\mathbf{p} \cdot \mathbf{n}_i > 0$, then $s(\mathbf{p}) = +1$, otherwise $s(\mathbf{p}) = -1$. The whole process is illustrated in Fig. 2.

To build a system of equations for plane fitting using a least squares fit, at least 3 data points are needed. In practice, a larger number is used depending of the density and noise of the data set. Two issues remain to be solved: First, the estimation algorithm has to compute consistent normal orientations. In our application, where the single 3D scans are taken by a mobile robot, the solution to this problem is

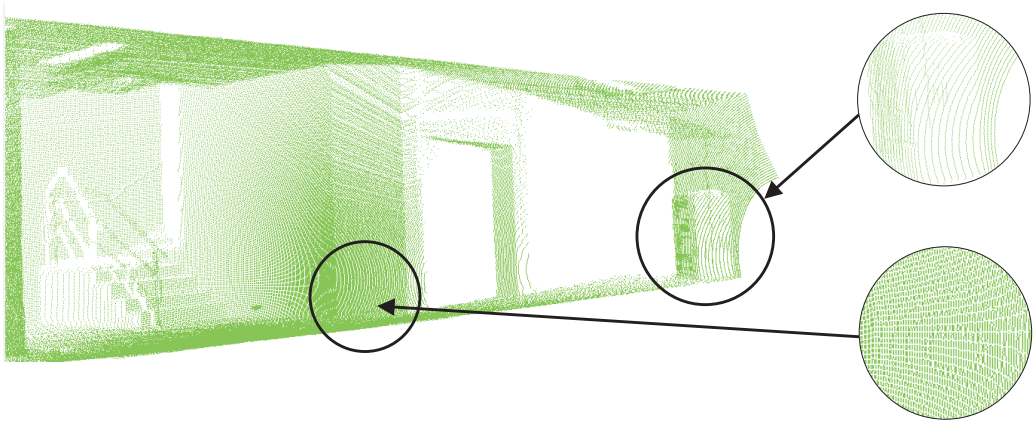

Fig. 3. An example of the varying point density in the used laser scans. The point density in the center of the scene is significantly higher than in the outer regions. In these areas the data points are aligned on lines, or arc-shaped.

Fig. 4. Influence of the normal estimation quality. Consistent normals result in connected surfaces (right) while inaccurate estimations produce holes in the triangle mesh (left).

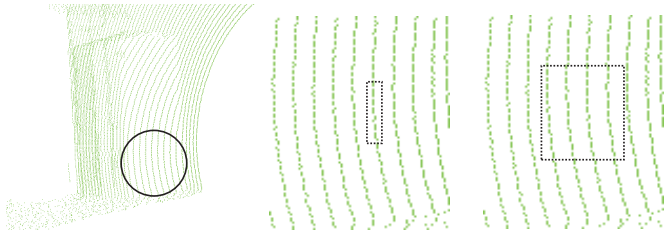

Fig. 5. Adaption of the number of nearest neighbors in regions with low point density. If k is too small, all data points will be on a straight line. In this case the fit of a plane will fully depend on the local noise of the data. To ensure a good fit, we analyze the shape of the bounding box.

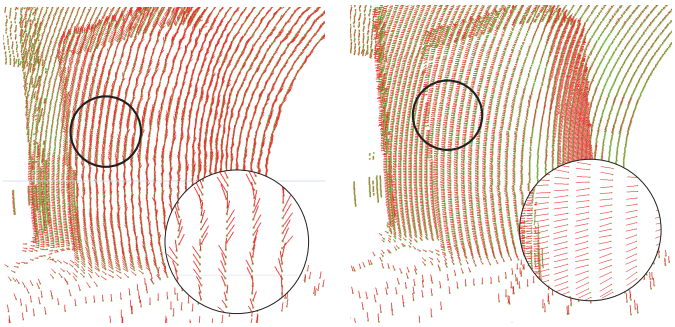

Fig. 6. Comparison between normal calculation without k adaption and interpolation (left) with the adaptive algorithm (right).

trivial: All normals are oriented towards the scanning position. A second problem is that the resulting distance function is not differentiable per se. The latter issue is not critical in this application, since the scanned environments are not necessarily smooth, due to sharp angles between, e.g., walls and floors. In fact, in robotics we exploit the ability to represent sharp features and the computational efficiency of Hoppe's approach.

The quality of the fitted tangent planes (and the corresponding surface normals and surface approximations) strongly depends on the number of chosen approximation points k (cf. Fig. 4). The choice of k is essential for the quality of the calculated normal. The smaller the value of k, the lower the needed processing time, since fever tree traversals are necessary. However, low k values are sensitive to noise. Higher values may compensate sensor noise in the approximation process, but they increase the processing time and might lead to wrong results, because sharp features will be "smoothed" out. Handling real 3D laser scanner data as input shows another difficulty: With increasing object distance the point density decreases. In the case of our tilted 2D laser scanner, this results in line or arc-shaped artefacts of data points at high distances (cf. Fig. 3). These artefacts will cause, in turn, incorrect results in the fitting process since the orientation of the calculated plane is solely dependent on local noise.

Thus we aim to find a k with a value as small as possible that still allows an accurate approximation. Therefore, we adapt k dynamically to the data density. To detect ill formed k-neighborhoods, we analyze the shape of their bounding boxes. Critical configurations will result in elongate bounding boxes. If we detect such a configuration, k is increased until this shape criterion is fulfilled. Since the laser scanner shows a lot of sensor noise, the resulting normals are still fluctuating to some degree, although their basic alignment is consistent. To reduce this effect, we average all normals with their neighbors. Fig. 6 shows the results of these optimization steps.

B. Extraction of Planar Surfaces

The aim of the second processing step is to detect planar areas in the mesh and to represent them as polygons. The triangles in the initial mesh are stored in a half edge representation. This data structure allows to efficiently detect all neighboring triangles of any triangle in the mesh. The simplification algorithm fuses patches in the mesh that are connected and share the same surface normal. The algorithm starts with an arbitrary triangle and recursively checks, if its surrounding triangles have a similar surface normals. The recursion is carried on until a bend in the surface is detected. The edge between such triangles marks a boundary of the initial surface. All these edges are collected and later fused using a line following algorithm to create an optimal polygonal representation (cf. Algorithm 1). Fig 7 shows the results of this process.

Algorithm 1 The mesh simplification algorithm. Faces within the mesh that have similar surface normals are detected. The border edges of these planar areas are fused to polygons.

```
function SIMPLIFY
    for all faces do
        current face ← visited
        FUSE(current normal, current face, currentList)
        borderLists ← currentList
        CREATEPOLYGON(border list)
        currentList ← empty
    end for
end function

function FUSE(start normal, current face, list of borders)
    current face ← visited
    for all neighbors of current face do
        angle ← start normal · neighbor normal
        if angle < ϵ and neighbor not visited then
            FUSE(start normal, neighbour, listOfBorders)
        else
            list of borders ← border edge to neighbor
        end if
    end for
end function
```

C. Semantic Labeling

Since architectural shapes of environments follow standard conventions arising from tradition or usage [4], we exploit this knowledge for semantic labeling of the polygonalization of indoor environments. The used knowledge describes general attributes of the domain, i.e., architectural features as plane walls, ceilings and floors.

The planarity constraint used to generate the polygonal representation is exploited to label the found surfaces by analyzing the surface normal orientations. In orthogonal scenes, the orientation of the normals is nearly discrete: Floor and ceiling normals point in the direction of the y-axis, walls are perpendicular to them. Using these considerations, we are able to label the found surfaces according to these categories. With textured rendering, realistic reproductions of the scanned environments can be created (cf. Fig. 8).

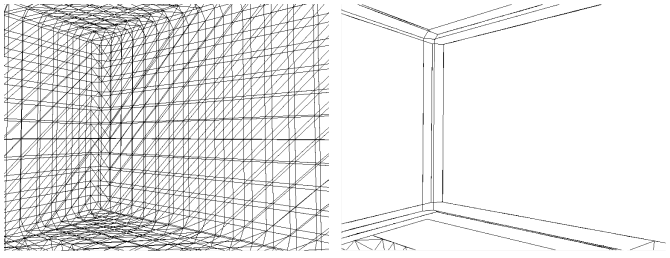

Fig. 7. Detailed view of a room corner. The triangles in the initial mesh are removed, only the borders of planar surfaces remain.

D. Implementation Considerations

The performance of the initial mesh generation procedure strongly depends on the efficiency of the k-neighbor search. The same problem is addressed during ICP scan matching in the SLAM 6D process. Furthermore the surface normal estimation for each point is a purely local operation and can be done independently for each data point, i.e., in parallel. Therefore we use search trees that are optimized for parallel queries. See [7], [12] for implementation details.

IV. Experimental Results

Fig. 9 shows an example of our automated polygonalization process. The left picture displays a registered point cloud generated from 12 single laser scans taken in an empty classroom. The middle picture is the initial triangle mesh based on the input data, created by our Marching Cubes implementation. The right picture shows the polygonal representation gained from the triangle mesh. The procedure has automatically extracted a polygonal representation of the large planes in the initial mesh without changing the geometry of the model. Note that in areas with high curvature the triangle representation is preserved, since these surfaces are not fused by our method.

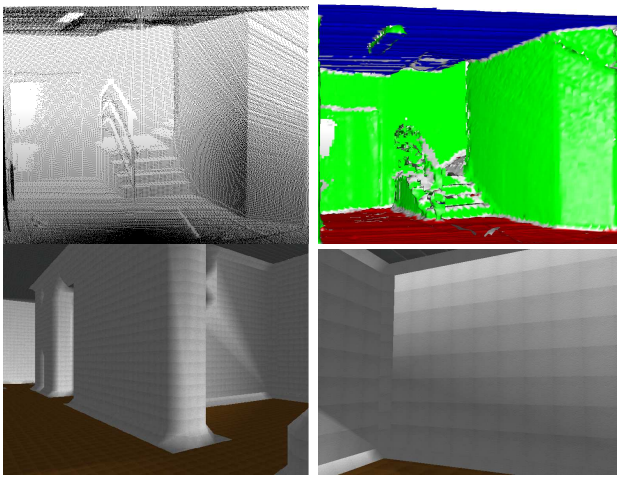

Fig. 8. Example of semantic labeling. Top row: The point cloud (left) was captured by the Kurt3D robot. In the reconstruction (right) the polygons are rendered with colors corresponding to their classification. Bottom: Another example. This time textures were added according to the surface classification.

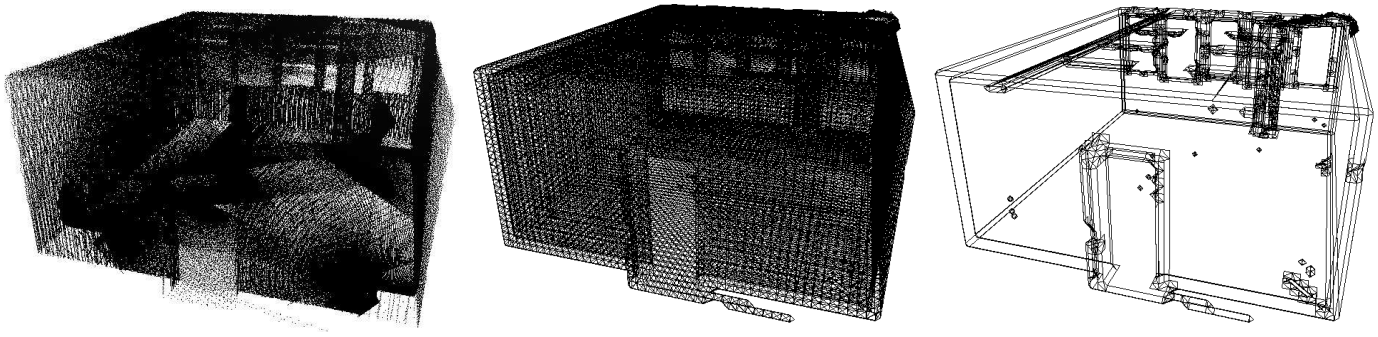

Fig. 9. The three steps of 3D map generation: The first step is to generate a consistent 3D point cloud of the robot's environment (left). This cloud is used to create a regular triangle mesh, using the Marching Cubes algorithm (middle). The third step is to detect planar surfaces in the mesh. The borders of these regions deliver a polygon representation of the environment (right).

TABLE I

RUN TIME AND COMPRESSION FOR THE DATASETS SHOWN IN FIG 8 (SINGLE SCAN) AND FIG. 9 (MULTIPLE REGISTERED SCANS).

Dataset	No. Points	Initial Faces	No. Polygons	Time
Single Scan	271,288	66,374	23,670	5.47 s
Multiple Scans	1,834,599	44,740	3,029	8.25 s

TABLE II

RUN TIME COMPARISON BETWEEN OUR OPTIMIZATION ALGORITHM AND OTHER MESH REDUCTION METHODS (REMOVAL OF THE SHORTEST EDGES AND USING QUADRIC ERROR METRICS [5]).

Dataset	Map Gen.	Shortest	Quadric	Compression
Single Scan	0.25 s	1.02 s	2.37 s	65 %
Multiple Scans	0.47 s	1.72 s	2.35 s	39 %

Table I displays the running times of the map generation procedure for the presented examples. The experiments were performed on an Intel Core2 Quad Q6600 with 4 GB RAM. Due to the parallel implementation we were able to achieve a load of nearly 100% on this machine. The normal estimation procedure scales well with the number of used threads. The first row shows the statistics for a single scan taken with our Kurt3D robot (see [13] for technical details), the second row the results for a set of 12 registered scans. In both cases, the number of initial polygons (triangles) was reduced considerably. Even in case of the large dataset (about 1.8 million points) the running time was lower than the time needed to acquire the scan data.

Additionally, we have compared our method to standard mesh reduction algorithms (see Table II). To create standard triangle meshes from our optimized representation, we used the OpenGL tesselator to re-triangulate the boundary polygons. Afterwards, we measured the time needed by the iterative methods to achieve an equal compression. In all tests, our method was faster, but did not change the geometry of the initial triangle mesh.

To evaluate the accuracy of the generated map, we have compared the reconstructed geometry shown in Fig. 9 with manual measurements in the original environment of ceiling height, wall width and height of a room and door width. The results are shown in Table III. The reconstructed values show a deviation of about 3 to 4 cm from the original values due to interpolation errors and noise in the original scans. Compared to the size of the mapped area these inaccuracies are negligible.

TABLE III

COMPARISON OF THE ORIGINAL AND RECONSTRUCTED GEOMETRIES.

	Ceiling	Width	Depth	Door Width
Original Dimensions	2.99 m	5.89 m	7.09 m	0.94 m
Reconstruction	2.96 m	5.85 m	7.06 m	0.90 m

V. APPLICATIONS

We have tested the usability of our 3D polygonal maps for localization purposes in different contexts. One example was the LiSA (Life Science Assistant) project. In this project methods for localization in 3D polygonal maps were developed and successfully applied. The used robot is equipped with several laser scanners at different orientations, that all use the same polygonal map. Localization is done using particle filters. Pose estimations for each sensor were generated based on raytracing in this model [14]. Fig. 9 shows the benefits of using several sensors: The left picture shows the positions of some of the laser scanners. The right picture compares the localization results between conventional 2D self localization (yellow) and the 3D approach (green). The pose estimation becomes more accurate when the additional information derived from the 3D polygonal map is used.

Furthermore we were able to apply a similar technique to realize real time 6D localization of a mobile robot with a PMD time-of-flight camera. For this experiment a pinhole camera model was used together with raytracing to generate an expectation of incoming sensor data under a given pose estimation. The expected data was then transformed to match the real data via ICP. This transformation was finally used to correct the initial estimation. [16] reports preliminary results.

These examples show that 3D maps can improve the results of localization and can be used with different kinds of sensors. Our current research focuses on tracking full 6D trajectories using IMUs and multi modal pose estimations.

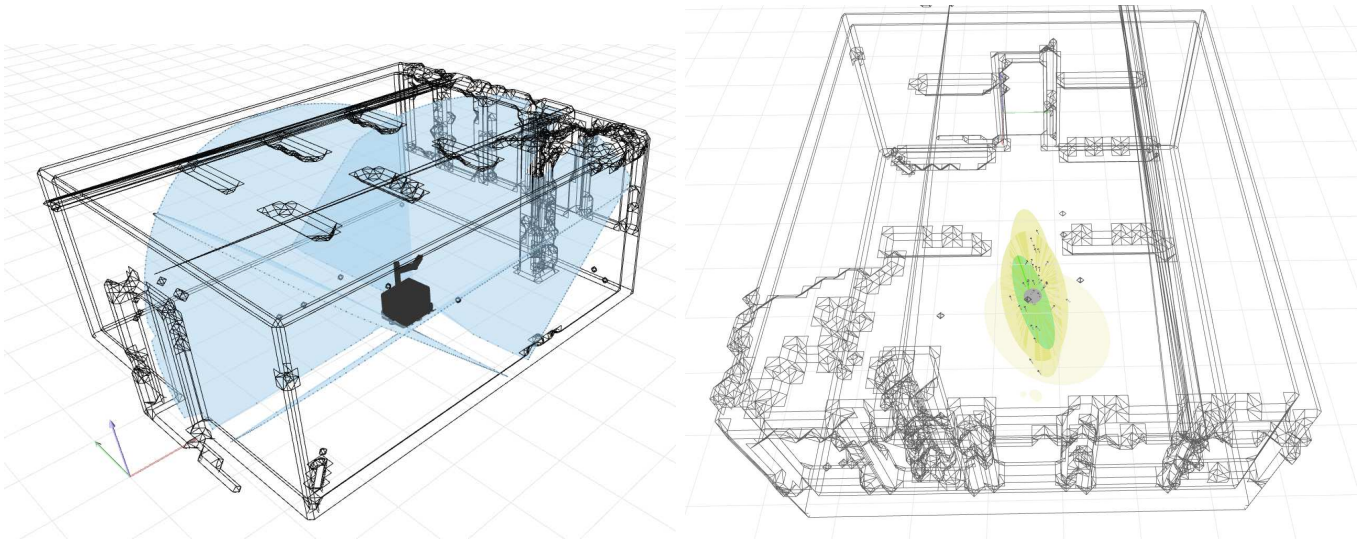

Fig. 10. Application example: Raytracing for localization in polygonal maps. The used robot us equipped with several laser scanners that point in different directions. A Raytracing technique is used to calculate a sensor model (left). The right figure shows the improvement of the self localization. The yellow area marks the estimated pose error without the use of the tilted scanners. The green area shows the result using the additional information from the other scanners. The localization improves considerably.

VI. Conclusion and Future Work

This paper has presented a novel approach to extract polygonal maps from a 3D point clouds. Such point clouds are the common output of a 3D mapping system. As several approaches from the field of computer graphics cannot be used in robotics, due to sensor noise and time constraints, we developed a simple and robust method to create practically usable 3D polygon maps. The mapping algorithm exploits the inherent scene structure of indoor environments. The planarity constraint is fulfilled in most robotic applications, including rescue scenarios (consider biological or chemical accidents).

The usability of the generated maps was shown for several localization tasks using the same map, but different kinds of sensors. Future work will focus on the following aspects:

- Fuse the purely mesh based approach with object recognition methods. Our simple classification according to planarity can be used to isolate non-planar regions in the scans, where potential objects of interest are located. Recognized objects (like chairs, tables, etc.) could then be replaced with pre-calculated models.
- Use the gained geometric information to improve the sensor data, e.g., fill laser shadows.
- Improve scan matching. Our 6D SLAM procedure currently depends on point-to-point references only. Using additional geometric information gained from the polygonalization could improve the scan matching results.

References

[1] M. Alex, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Computing and Rendering Point Set Surfaces In *IEEE Trans. Comp. Graphics Vis.*, 8(4), 2002

[2] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter. A Data Structure for the 3D Hough Transfrom for Plane Detection. In *Proc. 7th IFAC Symp. Intell. Aut. Vehicles (IAV 2010)*, September 2010.

[3] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. Globally consistent 3d mapping with scan matching. *J. Robotics and Auton. Syst.*, 65(2):130–142, 2008.

[4] R. B. Fisher. Applying Knowledge to Reverse Engeniering Problems. In *Proc. Int. Conf. Geom. Modeling and Process.*, pages 149 – 155, 2002.

[5] M. Garland and P. S. Heckbert. Surface Simplification Using Quadric Error Metrics. *Computer Graphics*, 31:209–216, 1997.

[6] A Golovinskiy, V. Kim, and Thomas Funkhouser. Shape-based Recognition of 3D Point Clouds in Urban Environments. In *Proc. International Conference on Computer Vision*, September 2009.

[7] M. Greenspan and M. Yurick. Approximate k-d Tree Search for Efficient ICP. In *4th Int. Conf. 3D Digit. Imaging and Modeling*, 2003.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[9] W. E. Lorensen and H. E. Cline. Marching cubes: A High Resolution 3D Surface Construction Algorithm. In *Proc. ACM SIGGRAPH*, 1987.

[10] F. Lu and E. Milios. Robot Pose Estimation in Unknown Envorinments by Matching 2D Range Scans. *J. of Intell. and Robotic Syst.*, 18(3):249–275, 1997.

[11] S. Melax. A Simple, Fast, and Effective Polygon Reduction Algorithm. *Game Dev. Mag.*, pages 47–49, Nov. 1998.

[12] A. Nüchter, K. Lingemann, and J. Hertzberg. 6D SLAM With Cached k-d Tree Search. In *Proc. 13th IASTED Intl. Conf. Robotics and Applications*, pages 181.–186, Würzburg, Germany, Aug. 2007.

[13] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM - 3D Mapping Outdoor Environments. *J. of Field Robotics*, 24(8/9):699 – 722, August/September 2007.

[14] S. Stiene and J. Hertzberg. Virtual Range Scan for Avoiding 3D Obstacles Using 2D Tools. In *Proc. 14th Intl. Conf. Advanced Robotics (ICAR)*, June 2009.

[15] S. Thrun. Robotic Mapping: A Survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[16] J. Wülfing, J. Hertzberg, K. Lingemann, A. Nüchter, S. Stiene, and T. Wiemann. Towards Real Time Robot 6D Localization in a Polygonal Indoor Map Based on 3D ToF Camera Data. In *Proc. 7th IFAC Symp. Intell. Auton. Vehicles (IAV 2010)*, Sep. 2010.