Estimation of Planar Surfaces in Noisy Range Images for the RoboCup Rescue Competition

Sarah Steinmetz
Active Vision Group
University of Koblenz-Landau
Universitätsstr. 1
56070 Koblenz, Germany
xena@uni-koblenz.de

Johannes Pellenz
Active Vision Group
University of Koblenz-Landau
Universitätsstr. 1
56070 Koblenz, Germany
pellenz@uni-koblenz.de

Dietrich Paulus
Active Vision Group
University of Koblenz-Landau
Universitätsstr. 1
56070 Koblenz, Germany
paulus@uni-koblenz.de

ABSTRACT

In this contribution we present a comprehensive method to extract connected planar polygons from a range image acquired by a laser range camera. The result is a boundary representation of the objects in the scene. The approximation of the detected planes by three–dimensional polygons can be used as a source for feature–based registration of sequential range images. We focus on the processing of range images that are segmented into planar regions. The polygon of each plane is extracted by incremental line fitting on the 2D contour of the segmented region projected onto the xy–plane followed by the propagation to the corresponding 3D–plane. We present a novel idea for joining these three–dimensional polygons. Due to sparsely distributed depth values of inclined planes and noise in areas of object edges, some planes cannot be segmented completely. Therefore object edges that actually represent one edge drift apart in 3D. Such edges are detected and joined. The direction in which each edge is moved, is determined by a confidence measure, depending on the slope of the plane.
We describe our experiments on range data of scenes containing planar as well as curved surfaces and give quantitative results. The estimates are compared to measurements that were manually taken. For cubic objects we compare the angles of the estimated polygons to the expected orthogonality. For approximately 70 planar surface patches we get an average error of 5 degrees. 83 percent of successfully segmented lines are joint correctly by the algorithm.

Keywords: 3D model reconstruction, boundary representation, range image processing, computer vision.

1 INTRODUCTION

Many publications propose and compare algorithms for the segmentation of dense range images into planar patches. An extensive survey of different techniques can be found in [HJBJ+96] and [CC05]. Here, we go further and process the segmented range images in order to extract a boundary representation of the scene from a single view. For our future goal, the autonomous indoor 3D map generation done by a robot at the RoboCup Rescue League, we are interested in the representation of the boundaries of planes as planar polygons. Such features may also be used for the registration of sequential range images. Concerning the problem of finding a high–level description of the segmented surfaces, there exist comparatively few techniques. In [BS92], after the segmentation of the range image, roof edges are built by the segments of the intersection lines of neighboured planes, whose 2D projection is verified by the roof edge contour pixels of these planes. Vertices result from the intersection of three planes. Step edges are found by 2D Hough clustering and a line tracing on the step edge contour pixels. The technique of intersection of surfaces can also be found in [FEF95], [Liu93] and [Koc96]. In section 3 we will see that this method is not applicable to our segmented range images due to to the noisy range data and unreliable orientation of some segmented planes. The approach in [HGB95] determines the relationship between planes (connected by edge or vertex) as well, by classifying the boundary pixels between regions as roof or jump edge pixels in an early processing step in order to determine which planes to intersect. However, it is demonstrated that one cannot fully trust in the correctness of these intersections, thus the authors use so–called ”glue patches” in order to handle the problem. With this method they even connect vertices that result from the intersection of more than three surfaces by a glue patch. We chose another approach to avoid faulty intersection of planes by determining common edges of planes after the extraction of the planar polygonal boundary of each plane. This higher–level description is more reliable than working on pixel level as we show in section 2.
Our approach expects a segmented range image as input. For the segmentation of the range images we use a modified reimplementation [Zin01] of the approach developed by Liu [Liu93]. The segmentation returns a

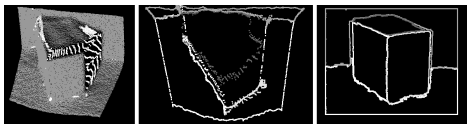

Figure 1: Left: outliers (*white points*) that are not assigned to any region, middle: the 3D contour coordinates for every region, right: orthographic projection of the contour coordinates onto the *xy*–plane

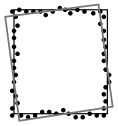

Figure 2: Incremental line fitting

region image that labels each pixel with the identifier of the region it belongs to and the parameters of the planes that have been fitted through the points of the regions. Some segmentation results are shown in Figure 7 in the lower left of each scene. After the segmentation there are still some erroneous outliers that actually belong to a region. The white points in the top plane of the box in the left image of Figure 1 represent such outliers. During the outlier elimination of the segmentation process the points remain marked as outliers due to their sparse distribution. The figure shows that range values in some visible areas, like the top of the displayed box, are as sparsely distributed as in areas of range values that represent artefacts. The white points between the front of the box and the wall represent artefacts. Particularly the areas of object edges are very noisy so that one can hardly distinguish a step edge from a roof edge by analyzing the neighbourhood of the border of a surface. Thus, in contrast to the approaches in [FEF95] and [HGB95], we do no initial classification of boundary pixels in order to determine the relationship between planes. Instead, we first determine the polygon for each plane (section 2), after that we analyze the relationship between edges in order to join common edges and vertices (section 3).

As imaging device we use a laser range camera, that provides dense range images with depth values in cartesian coordinates. Section 4 gives further information on the range images and the current results.

2 3D POLYGON EXTRACTION

2.1 2D boundary extraction

The first step is to determine the contour points of each region. For the boundary extraction we implemented the crack following algorithm [Kin97]. To identify the starting coordinate for this contour following method, the bounding box of the corresponding region is used. Starting from the bottom right corner the first pixel of the region is found by moving to the left. This procedure allows to assume that the contour starts with a corner pixel. The result is a sequential, counterclockwise list of the contour pixel coordinates for every region. The range image contains the *z*–coordinate for each pixel, thus the corresponding 3D contour coordinates are directly given, as shown in Figure 1.

2.2 Determination of 3D polygons

To determine the 3D polygon representing a plane, the best fitting lines through the contour points are needed. As depicted in Figure 1, the orthographic projection of the contour points onto the *xy*–plane represents the actual outline of the surface much better than the 3D coordinates. Thus the fitting is done in the 2D space, and the result is transferred to the 3D space.

In [NMTS05] several line extraction algorithms are compared. Incremental Line Fitting and Split and Merge are evaluated as best in speed and correctness. We implemented the incremental line fitting. With noisy data the standard incremental line fitting has a disadvantage: If a high residual threshold is chosen in order to avoid that a noisy point sequence is divided into many short segments, the resulting polygon is skewed, as depicted in Figure 2: the desired result is shown (*black rectangle*) overlayed to the result with high residual threshold (*skew, grey rectangle*).

Thus we extended the algorithm as follows: Every time when the next point is added to the fitting line, the current fitting error is compared to the fitting error of the previous step. If the new error is larger than the previous plus an offset, the current line is considered as getting worse. With the offset one can control the strictness of the judgement. When the current line finally exceeds the root–mean–square deviation threshold, not only the last bad point is put back, but all points since the fitting error became from bad to worse. Consequently one can choose a high fitting error threshold and fit lines through noisy data points, without fitting beyond corners. As line fitting we implemented an Orthogonal Distance Regression Fitting as described in [AV05]. The incremental line fitting results in a list of straight lines, the intersections of these subsequent lines represent the vertices of the desired polygon. The resulting 2D–polygon is transferred to 3D by computing the *z*–coordinate of the vertices using the parameters of the associated plane. If there are subsequent 3D–lines that are nearly parallel, the intersections are located outside of the polygon. Thus two lines that enclose an angle that is smaller than a threshold are merged by refitting the 2D–points of both lines. Figure 3 shows the result of the polygon extraction. The advantage of our approach is that we get closed linear contours (polygons) for each plane, thus we do not need any vertex refinement in this step.

3 JOINING OF POLYGONS

As shown in Figure 3 there are large gaps between polygons that should be actually connected. Especially the top of the box did ”move backwards”. Since some of such gaps are as large as distances between edges that do not belong together, a distance criterion in 3D space would be inappropriate, in order to detect which polygons (more precisely: which edges and vertices) should

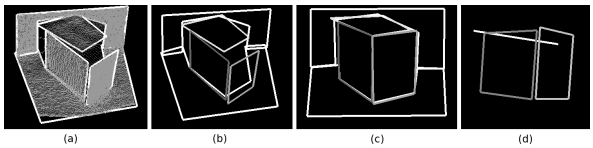

Figure 3: Result of the incremental line fitting on contour points of each plane. (a) with original range data, (b) extracted polygons, (c) orthographic projection onto the *xy*–plane, (d) the incoherent polygons of the box, view along the top plane

be joined. Again the analysis of the 2D orthogonal projection onto the *xy*–plane, compare Figure 3 (c), is advantageous. Criteria for joining two edges:

1. The 2D edges are nearly parallel.
2. The midpoints of the 2D edges are near.
3. The 2D edges have nearly equal length.
4. The 3D edges are nearly parallel.

But how to join two edges? Similar to the approaches mentioned in section 1, one could replace the concerned edges by an adequate part of the intersection line of the two planes that share the edge. Figure 3 (d) shows that this would cause an incorrect result concerning the shape of the box due to the erroneous orientation of the top plane.
Planes whose normal encloses a large angle with the optical axis of the camera, cannot be segmented completely due to sparse range data near edges, resulting in an erroneous orientation of the fitted plane. The idea is to adapt the orientation of such planes during the joining of edges. In order to be able to decide which plane should be adapted, a confidence value is assigned to every plane first. The chosen confidence measure is the angle between the normal of the plane and the *z*–axis, the smaller the angle the higher the confidence. A plane with a higher confidence (the "winner" plane) is allowed to influence the orientation of a less confidential plane (the "target" plane). Depending on the current state of the target plane, the influence of the winner plane decreases. States are:

1. FREE No edge of the target polygon has been joined yet. The orientation and position of the plane might be adjusted.

2. JOINED One edge of the target polygon has been joined. There is still one degree of freedom: the rotation around the joined edge.

3. FIXED The orientation of the target polygon is fixed, the vertices of edges that are not joined are allowed to move in its plane.

The algorithm is described in Figure 4, the operations in bold type are explained in the following.

3.1 Attracting the target plane

If the target polygon is FREE the influence of the winner polygon is unrestricted, as long as the adapted polygon keeps its planarity. The process of attracting the target plane is depicted in Figure 5. At first the target polygon is translated towards the winner edge. The translation vector $\boldsymbol{t}$ is the vector between that endpoint A_v of the target edge and that endpoint B_w of the winner edge that have the shortest distance. Then the target polygon is rotated in a way that the winner edge is part of the plane. According to Eulers's theorem, any rotation in 3D can be expressed as a rotation with respect to a single unit norm axis by an angle [TV98]. The rotation axis is the normalized vector $\boldsymbol{u}$ that is perpendicular to the direction vector $\boldsymbol{e}_\mathrm{w}$ of the winner edge and the direction vector $\boldsymbol{l}$ of the intersection line of the two planes. The rotation angle α is the acute angle enclosed by $\boldsymbol{l}$ and $\boldsymbol{e}_\mathrm{w}$. The rotation matrix $\boldsymbol{R}$ is computed as follows [TV98]:

$$\boldsymbol{R} = Id_3 \cos\alpha + (1-\cos\alpha)\cdot \boldsymbol{u}\boldsymbol{u}^\mathrm{T} + \sin(\alpha)\cdot[\boldsymbol{u}]_\times, \quad (1)$$

with identity matrix Id_3 and cross product matrix $[\boldsymbol{u}]_\times$. The rotation must not influence the already joined vertex, the intersection point of $\boldsymbol{u}$ and $\boldsymbol{e}_\mathrm{w}$. Therefore, before the polygon is rotated, the target polygon is moved to the origin using the negative position vector $\boldsymbol{t}_\mathrm{o} = -\boldsymbol{b}_\mathrm{w}$ of the joined vertex[1]. After the rotation the polygon is translated back. Using homogeneous coordinates the whole transformation can be expressed by a single transformation matrix $\boldsymbol{M} \in \mathbb{R}^{4\times4}$:

$$\boldsymbol{M} = (-\boldsymbol{T}_\mathrm{o})\,\boldsymbol{R}'\,\boldsymbol{T}_\mathrm{o}\,\boldsymbol{T} \quad (2)$$

$$\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \boldsymbol{t} \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{T}_o = \begin{pmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \boldsymbol{t}_\mathrm{o} \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

$$\boldsymbol{R}' = \begin{pmatrix} & & & 0 \\ & \boldsymbol{R} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4)$$

Each vertex $\boldsymbol{p}^i, i = 1,\ldots,n$, of the target polygon is expressed in homogeneous coordinates and multiplied by the matrix.

$$\left(p_1'^i, p_2'^i, p_3'^i, 1\right)^\mathrm{T} = \boldsymbol{M}\left(p_1^i, p_2^i, p_3^i, 1\right)^\mathrm{T} \quad (5)$$

The homogeneous coordinate is not changed, thus the new vertex $\boldsymbol{p}'^i$ is given by the first three components of the result.

$$\boldsymbol{p}'^i = \left(p_1'^i, p_2'^i, p_3'^i\right)^\mathrm{T} \quad (6)$$

[1] Position vectors of points are written in lower case.

Order planes descendant by confidence
Associate every plane with a descending list of its possible target planes
WHILE (there are still winner planes that have not been tested for joining yet)
  Take next winner plane p_w
  FOR every edge e_{w_i} of p_w
    WHILE (e_{w_i} not joined yet & there are still possible target planes)
      Take next target plane p_v
      WHILE (e_{w_i} not joined yet & there are still edges of p_v)
        Take next not joined edge e_{v_j} of p_v
        IF (e_{w_i} and e_{v_j} meet joining criteria)
          IF (p_v is FREE)
            attract p_v **to** e_{w_i}
            mark p_v as JOINED
          ELSE IF (p_v is JOINED)
            IF (e_{w_i} **and** e_{v_j} **share a common joined vertex**)
              turn p_v **round joined edge onto** e_{w_i}
              mark p_v as FIXED
            ELSE
              intersect p_w **and** p_v
              mark p_v as FIXED
          ELSE
            intersect p_w **and** p_v
            mark p_v as FIXED

Figure 4: Structogram: algorithm JoinPolygons

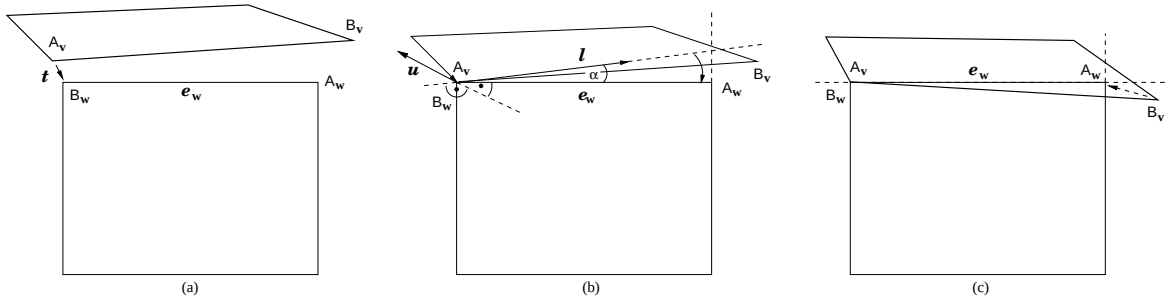

Figure 5: Attracting the target plane (a) Translation towards winner plane (b) Rotation onto winner edge (c) Substitution of the endpoint of the target edge by the endpoint of the winner edge.

After the transformation the target polygon and the winner edge lie in the same plane. The last step is to replace the second vertex B_v of the target edge with the corresponding endpoint A_w of the winner edge.
As this procedure changes the orientation of the target polygon, the execution is only allowed if the joining target edge is not too short with respect to the perimeter of the polygon. Otherwise the *Intersection* (see below) is performed. The condition prevents, that a small, error-prone edge propagates its possible error.

3.2 Turning of target plane

This part is executed if the winner polygon and the target polygon are already joined in a common vertex V_j, and if this vertex is the endpoint of a joined edge of the target polygon. Thus the target polygon is in state JOINED. The situation is depicted in Figure 6 (a). The target polygon is fixed at the joined edge with direction vector $\boldsymbol{e}_j$ but may rotate around it towards the winner polygon. As mentioned above, after the rotation the winner edge shall be part of the rotated target plane. For the computation of the rotation angle α we need a vector $\boldsymbol{l}_v$ in the target and a vector $\boldsymbol{l}_w$ in the winner

plane that are perpendicular to the rotation axis. These are computed by dropping a perpendicular from the free endpoints A_v and B_w of the edges that have to be joined to the line through V_j with direction vector $\boldsymbol{e}_j$.

$$t = \frac{(\boldsymbol{a}_v - \boldsymbol{v}_j) \cdot \boldsymbol{e}_j}{\|\boldsymbol{e}_j\|^2} \tag{7}$$

$$\boldsymbol{f} = \boldsymbol{v}_j + t \cdot \boldsymbol{e}_j \tag{8}$$

$$\boldsymbol{l}_v = \boldsymbol{a}_v - \boldsymbol{f} \tag{9}$$

$\boldsymbol{f}$ is the base of the perpendicular of A_v to the line. The computation of $\boldsymbol{l}_w$ is analogous. As the direction of the rotation depends on the direction of the rotation axis $\boldsymbol{u}$, it is computed by the normalized cross product $\frac{\boldsymbol{l}_v \times \boldsymbol{l}_w}{\|\boldsymbol{l}_v \times \boldsymbol{l}_w\|}$ The rotation angle α is the angle between $\boldsymbol{l}_v$ and $\boldsymbol{l}_w$. The rotation matrix $\boldsymbol{R}$ is computed as shown in formula 1. The final transformation is given by matrix $\boldsymbol{M} \in \mathbb{R}^{4\times4}$:

$$\boldsymbol{M} = (-\boldsymbol{T}_o)\,\boldsymbol{R}'\,\boldsymbol{T}_o \tag{10}$$

The transformation matrixes correspond to 4, with translation vector $\boldsymbol{t}_o = (-1) \cdot \boldsymbol{v}_j$ for the translation of the target plane to the origin. After the transformation of each vertex of the target polygon, the free endpoint A_v of the target edge is replaced by the corresponding endpoint B_w of the winner edge.

Intersection The intersection causes the adaption of both, the target edge and the winner edge. The edges are replaced by a segment of the intersection line of the concerned planes. The segment is the line composed by the intersection points of the preceding and the following edge of the winner edge with the target plane.

The result of the process is a boundary representation of the scene. The geometry is given by the 3D polygons and the according plane parameters, the topology is given by the connection of edges and trihedral vertices.

4 EXPERIMENTS AND RESULTS

For the experiments we used a laser range camera of Daimler-Benz Aerospace, that provides dense range images. We used a resolution of 320×240 pixels and a measuring range of 90 to 300 cm, the distance resolution is about two percent of the measuring range, as evaluated in [APZN01]. We tested the method with artificial indoor scenes, 8 images that show objects with planar surfaces (image number p01 to p08) and 5 images that show also objects with curved surfaces (image number m01 to m05). Figure 7 shows several results. The top left chart shows the original range image in 3D space, the bottom left displays the segmentation result and the right chart shows the final result. All images result from the same parameter settings.

Image	Edges to join		Joined	
	before	after	correct	wrong
	polygon extraction			
p01 (a)	7	7	7	0
p02	4	4	4	0
p03	7	5	4	0
p04	7	7	5	0
p05 (b)	7	7	7	0
p06	2	2	2	0
p07	7	3	3	0
p08 (c)	8	4	4	0
m01 (d)	6	5	3	1
m02	6	6	3	0
m03	2	2	0	0
m04	9	6	6	0
m05	6	4	3	0
Total	78	62	52	1

Table 1: Evaluation of joining of edges of polygons. The letters in brackets in the first column refer to the images in figure 7

Image	Trihedral vertices to join		Joined
	before	after	
	polygon extraction		
p01 (a)	2	2	2
p02	1	1	1
p03	1	0	0
p04	2	1	1
p05 (b)	2	2	1
p06	0	0	0
p07	2	0	0
p08 (c)	1	0	0
m01 (d)	1	1	1
m02	2	2	1
m03	0	0	0
m04	2	1	1
m05	0	0	0
Total	16	10	8

Table 2: Evaluation of joining of vertices of polygons. The letters in brackets in the first column refer to the images in figure 7

Table 1 and table 2 list the results of the joining of edges. The evaluation shows that some edges and accordingly trihedral vertices are lost due to under-segmentation, but 83.87% of the edges that remain to join after the extraction of polygons for each plane are merged correctly. Image 7(a) shows an example where all edges could be joined and the shape of the box is well approximated. Especially the top of the box, that had shown an erroneous orientation after the segmentation, has been adjusted properly. Table 3 shows the reconstructed angles between planes that are known to be perpendicular. The mean difference to the optimum of 90° averages 5.34°.

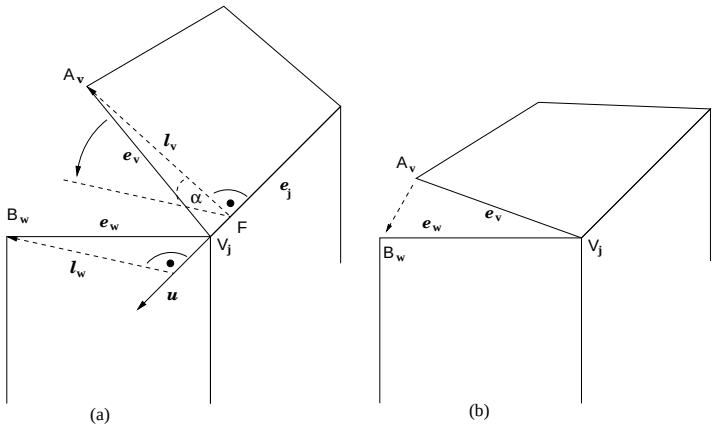

Figure 6: Turning of target polygon around the joined edge $\boldsymbol{e}_j$ (a) Turning of the target polygon onto the winner edge (b) Substitution of the free endpoint of the target edge.

Image	Number of orthogonal planes	Average angle between planes	Diff. to 90°
p01 (a)	7	86.94	3.06
p02	7	84.14	5.86
p03	7	83.99	6.01
p04	10	85.54	4.46
p05 (b)	7	85.47	4.53
p07	1	83.90	6.10
p08 (c)	5	83.32	6.68
m01 (d)	10	84.00	6.00
m02	7	83.49	6.51
m03	4	82.65	7.35
m04	3	87.27	2.73
Total	68	84.66	5.34

Table 3: Evaluation of angles between orthogonal planes. Angles are given in degree. The letters in brackets in the first column refer to the images in figure 7

Image 7(b) shows an artefact caused by the proceeding during intersection: In this case the front plane of the box has been the winner polygon and the ground the target polygon. The preceding edge of the bottom edge of the front had been a short sloped edge and this skewness is optically enforced because this edge becomes longer due to the intersection process. Image 7(c) shows the already mentioned problem of undersegmentation concerning planes, whose normal has a large angle to the optical axis. Nevertheless all edges have been joined correctly in these images. Our experiments confirm that the joining criterion (section 2) should be adapted. The current criteria are not strict enough to ensure that two edges are only joined if they really represent a common edge, as shown in image 7(d). The slope of the top plane of the box is wrong as its rear edge had been joined erroneously with its shadow edge that belongs to the wall plane. The image 7(d) shows an example with objects with curved surfaces. These are approximated by several planar patches. Overall the results are promising, especially the joining and the adjustment of the orientation of polygons works very well.

We also tested the method on data that was acquired with a 3D laser range finder which is mounted on our mobile system. The 3D–scanner consists of a Hokuyo 2D range finder that can be turned by a servo [PDMP06]. It covers a sphere of $240^\circ \times 360^\circ$ degrees with a resolution of $0.352^\circ \times 0.9^\circ$. The scanner can detect objects in distances up to 4000 mm. It is lightweight and small and therefore very suitable for a mobile robot. For the experiments, range data of an almost empty room was captured (see figure 8). The only objects in that room were benches and some lamps that are attached to the wall. From the complete sphere, a window of about $139^\circ \times 127^\circ$ was extracted and projected on a plane of 640×480 pixels (see figure 9). For each pixel in the plane, the ϕ and θ of the angle from the camera center was calculated and the corresponding 3D–coordinate was determined by a linear interpolation of the 4 closest laser range measurements. The result of this transformation is a range image comparable to the data that comes from the laser camera, but with a much larger field of view (the field of view of the laser range camera is only 42° in horizontal and 32° in vertical direction). This results in a high resolution of 3D–points along the z–axis even if the object is located in the area above and laterally of the laser scanner.

Therefore the processing of the images created by the laser scanner requires a few adaptions of the algorithm. The reason is that the projection of the computed 3D–data onto the xy–plane leads to a representation that is not adequate for the incremental line fitting process. In contrast to the images of the laser camera the new 3D–data may contain planes whose normal is nearly orthogonal to the principal axis and whose projection onto

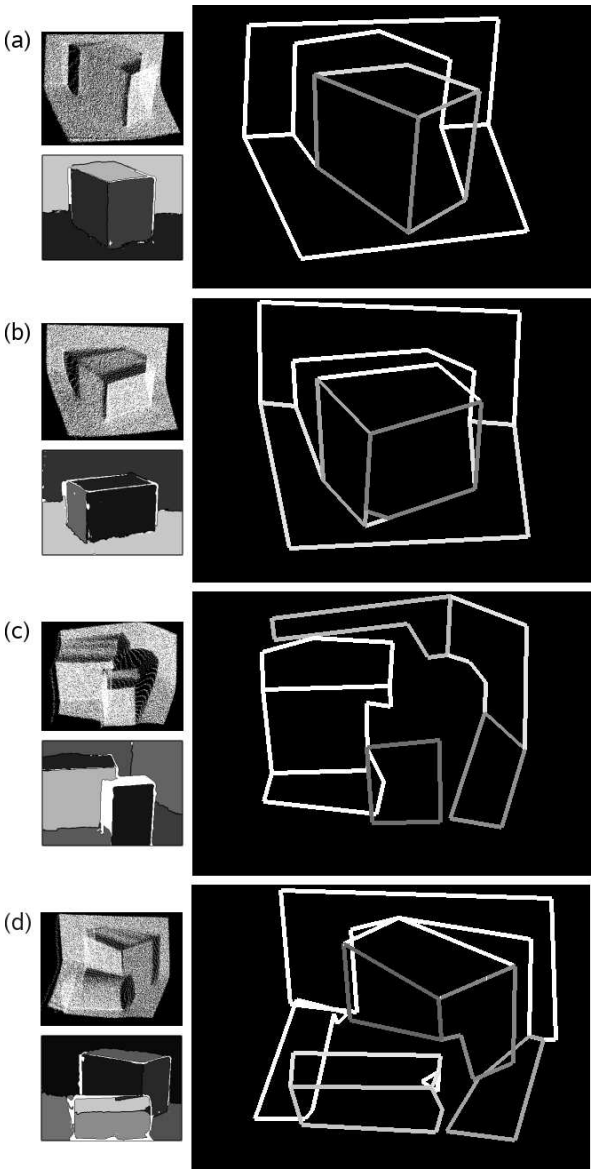

Figure 7: Some results of the whole process. Top left chart: original range image in 3D space, bottom left chart: segmentation result, outliers are depicted as white points, right chart: final result

the xy–plane results in collapsed shapes with nearly no area. Thus the incremental line fitting and the transfer of the 2D–polygons onto the according 3D–plane may fail. The propagation of a 2D–vertex to the corresponding plane could result in a point in the infinite. The new approach is that the incremental line fitting is not always processed with the projection onto the xy–plane but with the projection onto that coordinate plane whose direction of projection is closest to the direction of the normal of the plane. Further the second joining criterion, that checks if the midpoints of the 2D–edges are close enough, is now performed in 3D–space.

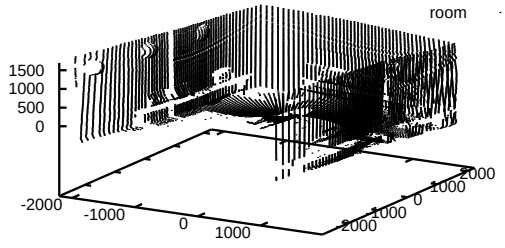

Figure 8: Point cloud of the scanned room (acquired by the 3D laser scanner; without the ceiling)

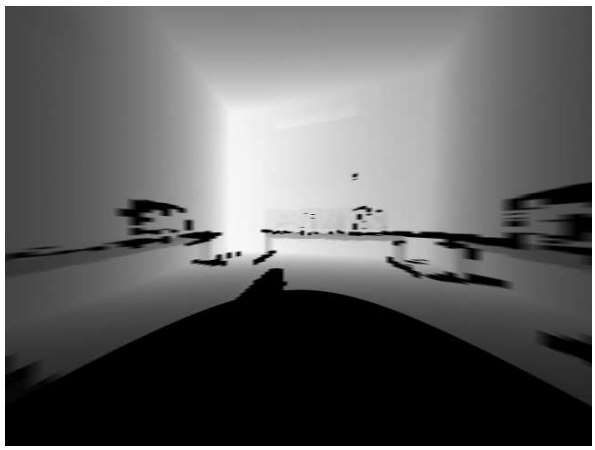

Figure 9: Plane with depth data as grey values. The black artefacts represent areas where the laser scanner did not receive an echo

Figure 10 displays the result for the described scene. The example demonstrates that smaller artefacts (also inside of regions) are suppressed. The qualitative analysis shows that the slightly modified algorithm works also well with the range data from the laser scanner.

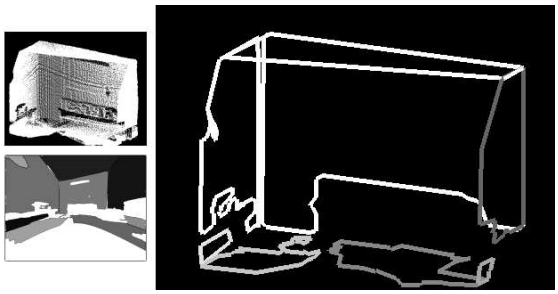

Figure 10: Result using an image created with the laser range scanner. Top left chart: original range image in 3D space, bottom left chart: segmentation result, outliers are depicted as white points, right chart: final result

5 CONCLUSION

We presented a method to extraxt a boundary representation from dense range images based on their segmentation into planar surfaces. Using a confidence measure the polygons are joined by edges and trihedral vertices. In addition the orientation of less reliable polygons is adjusted during this process. The result is a high level description of the scene that contains its geometry and topology. In the future we want to enforce the joining criterion, in order to assure that different edges are not joined. The technique will be used for data from a 3D laser range finder which is mounted on our mobile robot "Robbie" at the RoboCup Rescue championship next year.

REFERENCES

[APZN01] U. Ahlrichs, D. Paulus, T. Zinsser, and H. Niemann. Segmentation of range images. In S. Donati, editor, *Optoelectronic Distance / Displacement Measurements and Applications*, pages 416–421, Pavia, Italy, 2001.

[AV05] H. Aceves and H. Velazquez. Scalings between physical and their observationally related quantities of merger remnants. *Revista Mexicana de Astronomia y Astrofisica*, (41):523–532, 2005.

[BS92] Suchendra M. Bhandarkar and Andreas Siebert. Integrating edge and surface information for range image segmentation. *Pattern Recognition*, 25(9):947–962, 1992. `http://dx.doi.org/10.1016/0031-3203(92)90060-V`.

[CC05] C. Chao Chen. A survey of techniques on range image segmentation and registration. Ph.D. Second-level Exam Part I, `http://web.cs.gc.cuny.edu/~cchen/`, 2005.

[FEF95] A. W. Fitzgibbon, D. W. Eggert, and R. B. Fisher. High-level cad model acquisition from range images. In *University of Edinburgh*, Department of Artificial Intelligence, 1995. University of Edinburgh.

[HGB95] A. Hoover, D. B. Goldgof, and K. W. Bowyer. Extracting a valid boundary representation from a segmented range image. 17(9):920–924, 1995.

[HJBJ+96] A. Hoover, G. Jean-Baptiste, X. Jiang, P. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, and R. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Trans. on PAMI*, 18(7):673–689, 1996.

[Kin97] V. Kindratenko. *Development and Application of Image Analysis Techniques for Identification and Classification of Microscopic Particles*. PhD thesis, University of Antwerp, Department of Chemistry, 1997.

[Koc96] R. Koch. Surface segmentation and modelling of 3-d polygonal objects from stereoscopic image pairs. In *International Conference on Pattern Recognition*, volume 1, pages 233–237, 1996.

[Liu93] H. Liu. *Oberflächenbasierte Segmentierung von Tiefenbildern*, volume 36 of *Dissertationen zur Künstlichen Intelligenz*. Infix Verlag, St. Augustin, Germany, 1993.

[NMTS05] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. *IEEE International Conference on Intelligent Robots and Systems*, 2005.

[PDMP06] Johannes Pellenz, Christian Delis, Ioannis Mihailidis, and Dietrich Paulus. Low-cost 3D-Laserscanner für mobile Systeme im RoboCup Rescue Wettbewerb. *9. Anwendungsbezogener Workshop zur Erfassung, Modellierung, Verarbeitung und Auswertung von 3D-Daten*, 2006.

[TV98] E. Trucco and A. Verri. *Introductory Techniques for 3–D Computer Vision*. Prentice Hall, New York, 1998.

[Zin01] T. Zinsser. Oberflächensegmentierung in Tiefenbildern. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, http://www.informatik.uni-erlangen.de, 2001.