Fast Range Image Segmentation and Smoothing Using Approximate Surface Reconstruction and Region Growing*

Dirk Holz and Sven Behnke

Autonomous Intelligent Systems Group, University of Bonn, Germany
{holz,behnke}@ais.uni-bonn.de

Abstract. Decomposing sensory measurements into relevant parts is a fundamental prerequisite for solving complex tasks, e.g., in the field of mobile manipulation in domestic environments. In this paper, we present a fast approach to surface reconstruction in range images by means of approximate polygonal meshing. The obtained local surface information and neighborhoods are then used to 1) smooth the underlying measurements, and 2) segment the image into planar regions and other geometric primitives. An evaluation using publicly available data sets shows that our approach does not rank behind state-of-the-art algorithms while allowing to process range images at high frame rates.

1 Introduction

As robots and autonomous systems move away from laboratory setups towards complex real-world scenarios, both the perception capabilities of these systems and their abilities to acquire and model semantic information must become more powerful. A key issue is the extraction of semantic information from sensory data and its decomposition into parts of interest that are relevant for the tasks of the robot. For mobile manipulation in domestic environments, for example, the perception of objects and their surroundings is a key prerequisite. A common approach [1] in 3D perception is to

1. detect horizontal support planes,
2. extract and cluster points on top of these planes, and
3. perform further processing, e.g., recognizing, classifying or tracking of the found clusters.

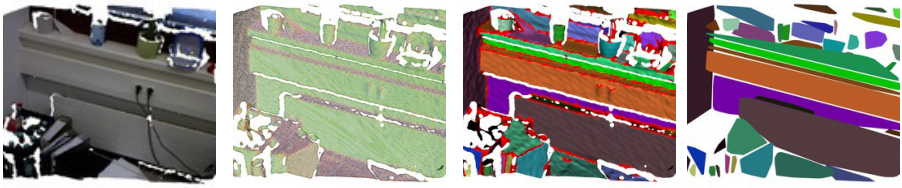
Here, one of the fundamental problems is to segment the 3D data into planes and other geometric primitives—or regions of local surface continuity in general.

In this paper, we address the problem of segmenting range images and organized point clouds in real-time on domestic service robots. The central idea of our approach is to approximately reconstruct the surface and segment the range image by growing regions using the resulting local mesh neighborhoods. By means of easily exchangeable components, our generalized region growing approach allows for different region

* This research has been partially funded by the FP7 ICT-2007.2.2 project ECHORD (grant agreement 231143) experiment ActReMa.

S. Lee et al. (Eds.): Intelligent Autonomous Systems 12, AISC 194, pp. 61–73.
springerlink.com

models (e.g. planes) that are segmented in the data. We present models for segmenting planes, regions of local surface continuity, and simple geometric primitives at high frame rates (see Fig. 1).



(a) Input cloud (b) Triangle mesh (c) Segmentation (d) Polygonalization

Fig. 1. Example of surface reconstruction and (plane) segmentation on a RGB-D point cloud: (a) input cloud; (b) constructed mesh; (c) result of segmenting planes in the mesh (red points are assigned to multiple planes); (d) polygonalization as a collection of alpha shapes. Using the Microsoft Kinect camera at a resolution of 160×120, we can compute full polygonalizations with roughly 30 Hz on a standard dual-core notebook.

Furthermore, we use the same mesh neighborhoods to 1) efficiently compute local surface normals and curvature estimates, as well as 2) smooth both the 3D measurements and the computed normals using a bilateral filter.

This paper is organized as follows: After giving a brief overview on related work in range image and 3D plane segmentation in Section 2, we present our approach in Section 3 and discuss how to detect geometric primitives using initial segmentations. We use multiple data sets for evaluating the efficiency and robustness of our approach and summarize the results in Section 4.

2 Related Work

Research on computer and robot vision produced a wide variety of approaches to range image segmentation—and plane segmentation in particular. Hoover et al. [2] compiled a survey and performed an evaluation of early work. Three different types of approaches can be distinguished according to the underlying working principle: methods using random sample consensus (RANSAC), 3D Hough transforms, and region growing. In the context of segmenting 3D laser range scans, another popular approach is to first detect lines in planar cuts, and to merge neighboring lines into local plane patches (see Vosselman et al. [3] for an overview).

2.1 Segmentation Based on Sample Consensus

RANSAC-based approaches try to find models for geometric primitives that best explain a set of points and the set of inliers supporting it. For segmenting a complete range image, Lee et al. [4] sequentially remove inliers from the original data set, and

continue the segmentation with the residual points. Silva et al. [5] first identify connected regions and apply RANSAC region-wise. Gotardo et al. [6] compute an edge map for pre-segmentation and use a variant based on the M-estimator sample consensus (MSAC) to fit model parameters.

Another efficient solution to segmenting even unorganized point clouds and detecting simple geometric primitives such as planes, cylinders and spheres has been proposed by Schnabel et al. [7]. They decompose unorganized point clouds using an octree subdivision and apply RANSAC only to subsets of the original point cloud.

In previous work [8], we adapted the perception scheme from Section 1 as well as the techniques from [1] and [7], and made them applicable to the measurements of time-of-fight (ToF) cameras. We presented techniques to cope with the specific error sources of the cameras, and to speed up processing by exploiting the image-like data organization. After detecting the most dominant plane, we applied the octree-based primitive detection of Schnabel et al. [7] only to already extracted and segmented points above that plane. In [9], we further speeded up the segmentation process by using integral images for computing local surface normals more efficiently, and using the index neighborhood underlying the 3D data to extract and track segments of points and object candidates, respectively. The overall approach is applicable in real time on a Microsoft Kinect camera and has been used for real-time object tracking and grasp planning [10].

2.2 Hough-Based Plane Segmentation

The Hough transform is the de-facto standard for finding lines and circles in 2D images. Various extensions to 3D exist that try to find, respectively, planes and maxima in histograms over the possible space of plane orientations and distances. For an overview, and an evaluation for Hough-based segmentation approaches, we refer to the works of Vosselman et al. [3] and Borrmann et al. [11].

RANSAC- and Hough-based segmentation share a common disadvantage. Points belonging to the same segment do not necessarily lie on connected components. Both approaches will merge plane segments if they share a common orientation and distance to the origin. In addition, Hough-based segmentation may suffer from discretization effects.

In [9], we present a fast plane segmentation approach that uses a similar parameter space as the Hough transform. We pre-cluster points and segment planes first in normal space and then, for each cluster, in distance space to obtain individual planes. We compensate for discretization effects by conducting a post-processing step in which neighboring segments are merged if their parameters do not considerably deviate. Still, unconnected planar patches may get merged into the same cluster.

2.3 Segmentation Using Region Growing

The idea of region growing-based segmentation is to exploit the image-like data structure. Hähnel et al. [12] connect neighboring points in 3D laser range scans to a mesh-like structure. The scans are then segmented recursively by merging connected patches that are likely to lie on the same planar surface. Poppinga et al. [13] apply the same approach to Time-of-Flight cameras and re-formulate the algorithm in an incremental

fashion. They grow planar regions by adding neighboring points whose distance to the plane lies below a threshold. Centroid and covariance matrix for estimating the plane parameters are thereby incrementally updated.

Here, we follow a similar approach. Instead of incrementally computing the covariance matrix, however, we compute the normals for all points beforehand and simply average local surface normals to obtain an estimate of the plane's normal. That is, we only store and incrementally update the centroids in both Cartesian and normal space.

Other popular region growing-approaches to range image segmentation make use of local surface curvature. Regions are grown until points with a considerably larger curvature are reached. Just like Gotardo et al. [6] for RANSAC-based segmentation, Harati et al. [14], first compute an edge map to find connected regions of local surface continuity. Rabbani et al. [15] approximate local surface curvature by first fitting planar segments to local point neighborhoods and then computing, for each point, the distance to that plane. Recently, Cupec et al. [16] followed a similar approach. They first apply 2.5D Delaunay triangulation on a range image to obtain an initial triangular mesh and then use the maximum distance of an examined point to all triangles in a region to determine whether or not the point is added.

Here, we deduce a surface reconstruction directly from the image-like data structure, and use the local ring neighborhood around vertices to 1) efficiently compute local surface normals and curvature estimates, and 2) efficiently smooth the depth measurements using a bilateral filter. Our framework allows for using different types of models for region growing, including the approaches of Poppinga et al. [13], Harati et al. [14], Rabbani et al. [15] and Cupec et al. [16], as well as our fast approximation (see Section 3).

3 Fast Mesh Construction and Segmentation

In this section, we describe our approaches to approximate surface reconstruction and segmentation based on region growing. The overall processing pipeline is composed of the following components:

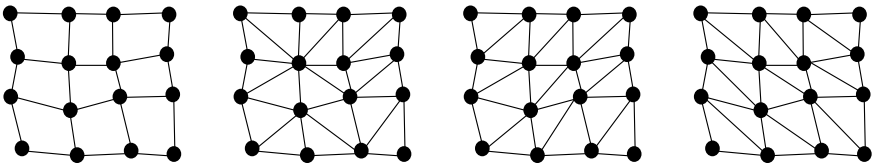
1. Deduce approximate mesh from image neighborhood.
2. Use mesh neighborhood to compute approximate local surface normals and curvature estimates.
3. Bilateral filtering to smooth both points and normals.
4. Segmentation based on region growing.

All components described in the above list use the same fixed neighborhoods. These neighborhoods come either directly from the mesh structure or can be pre-computed using a variety of search trees, if the input data is unstructured.

3.1 Exploiting Structure for Fast Approximate Meshing

The central idea of our surface reconstruction approximation is to deduce the desired mesh structure directly from the image-like organization of measurements. In fact, the

algorithms presented in the following could easily be applied on local index neighborhoods in range images. However, an approximate mesh allows for 1) applying a wide variety of sophisticated algorithms from the field of computer graphics, as well as for storing certain edge weights, e.g., the difference vectors for integral image-based normal computation from our previous work [9].



(a) Quad (b) Adaptive (c) Left cut (d) Right cut

Fig. 2. Fast approximate meshing using a quad mesh (a) and different triangulations (b-d). Compared to the adaptive approach (b), triangulations using only left cuts (c) or only right cuts (d) can be obtained slightly faster.

We traverse a given range image R once and check for every point $\mathbf{p}_i = R(u,v)$:

- if $R(u,v)$ and its neighbors $R(u,v+1)$, $R(u+1,v+1)$, and $R(u+1,v)$ (in the next row and the next column) are valid depth measurements, as well as
- if all edges between $R(u,v)$ and these three neighbors are not occluded.

The first check is necessary because of the structure in the sensory data that we exploit. If the sensor cannot acquire a valid depth measurement for a certain pixel, it has to store an invalid one, in order to keep the structure organized.

The latter occlusion checks can be easily done by examining the difference vectors between $\mathbf{p}_i$ and its three neighbors. If it falls into a common line of sight with the viewpoint from where the measurements have been taken, one of the underlying surfaces occludes the other. The condition for the validity of an edge between point $\mathbf{p}_i$ and its neighbor $\mathbf{p}_j$ can be formulated as

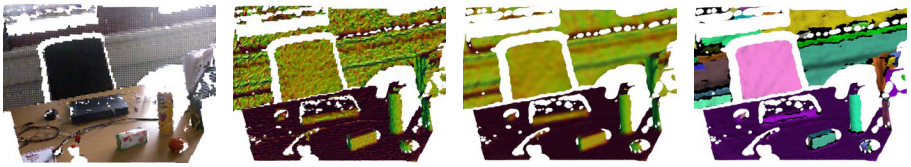
$$valid = \left(\left(\left| \mathbf{p}_i \cdot \mathbf{p}_j \right| \leq \cos \varepsilon_\theta \right) \wedge \left(\|\mathbf{p}_i - \mathbf{p_j}\|^2 \leq \varepsilon_d^2 \right) \right), \tag{1}$$

where ε_θ and ε_d denote maximum angular and length tolerances, respectively.

If all checks are passed, $R(u,v)$ and its neighbors are used to extend the so far built mesh. Otherwise, holes arise. Referring to Fig. 2, we distinguish four types of meshes:

1. Quad meshes are formed by connecting pixel $R(u,v)$ to $R(u,v+1)$, $R(u+1,v+1)$, and $R(u+1,v)$.
2. Fixed left and right cut meshes are formed by cutting quads either from top right to bottom left (left cut) or from top left to bottom right (right cut).
3. Adaptive triangulation cuts the quad along the diagonal that has a smaller length.

For triangulations, a single invalid neighbor causes that only one triangle is added. After construction, we simplify the resulting mesh by removing all vertices that are not used in any polygon. An example triangulation is shown in Fig. 3(b).



(a) Input cloud (b) Initial mesh (c) Filtered mesh (d) Segmented mesh

Fig. 3. Approximate surface reconstruction, bilateral filtering and segmentation on an example point cloud (a). The filtered mesh (c) is considerably smoother than the initially approximated mesh (b) and allows for cleanly segmenting regions of local surface continuity (d). The meshes in (b) and (c) are colored w.r.t. the vertices' normal orientation (mapped to RGB space, object-space normal map). In (d) segments are colored randomly.

3.2 Fast Computation of Surface Normals and Curvature

We compute the local surface normal $\mathbf{n}_i$ for point $\mathbf{p}_i$ as the weighted average of the plane normals of the faces surrounding $\mathbf{p}_i$. Using the cross product between the difference vectors of the bounding vertices to compute the face normals, and choosing the weights to be proportional to the area of triangles, removes the need of normalizing the face normals beforehand. Thus, we can obtain $\mathbf{n}_i$ as:

$$\mathbf{n}_i = \frac{\sum_{j=0}^{N_T} (\mathbf{p}_{j,a} - \mathbf{p}_{j,b}) \times (\mathbf{p}_{j,a} - \mathbf{p}_{j,c})}{\left\| \sum_{j=0}^{N_i} (\mathbf{p}_{j,a} - \mathbf{p}_{j,b}) \times (\mathbf{p}_{j,a} - \mathbf{p}_{j,c}) \right\|}, \tag{2}$$

where $\mathbf{p}_{j,a}$, $\mathbf{p}_{j,b}$ and $\mathbf{p}_{j,c}$ form triangle j. In the actual implementation, we simply iterate over the faces, compute the difference vectors and their cross product, and add them to the normals of the involved points. Finally, we normalize all point normals at once. An example for computed local surface normals (color coded) can be seen in Fig. 3(b).

3.3 Bilateral Filtering

Naturally, sensor measurements are affected by noise. Since this noise can hinder further processing like segmentation, we apply a bilateral filter for smoothing both the points and their normals while preserving edges in the sensed geometric structures. Again, instead of searching, we directly extract a point's neighborhood from the mesh. That is, we filter both a point $\mathbf{p}_i$ and its normal $\mathbf{n}_i$ over its 1-ring-neighborhood N_i, i.e., all points that are directly connected to $\mathbf{p}_i$ by an edge in the mesh:

$$\mathbf{p}_i = \sum_{j \in N_i} w_{ij} \mathbf{p_i} \Big/ \sum_{j \in N_i} w_{ij}, \qquad \mathbf{n}_i = \sum_{j \in N_i} w_{ij} \mathbf{n_i} \Big/ \sum_{j \in N_i} w_{ij}, \tag{3}$$

$$w_{ij} = \underbrace{e^{\|\mathbf{p}_i - \mathbf{p_j}\|}}_{\text{distance term}} \underbrace{e^{\|\mathbf{n}_i - \mathbf{n}_j\|_1}}_{\text{normal term}} \underbrace{e^{(I_i - I_j)/c_I}}_{\text{intensity term}}, \tag{4}$$

where the optional intensity term is only evaluated for colored point clouds and range images where also an intensity image is available. The normalization constant c_I is used to scale the intensity differences to lie in the interval $[0, 1]$. An example of filtering an input mesh can be seen in Fig. 3(c).

3.4 Region Growing-Based Segmentation

Despite the generalization over different neighborhood searches and region models, the implementation of our segmentation algorithm does not considerably deviate from other region growing algorithms in the literature. Given a set of seed points (and a priority queue of seeds),

Outer loop, until all points are processed:
1. we iteratively select the next seed point,
2. initialize the region model of interest, and
3. put the seed point onto the empty processing queue.

Inner loop, while the processing queue is not empty:
4) We take the next point from the processing queue,
5) check its compatibility with the region model,
6) and add it in case of compatibility.
7) We add the point's neighbors to the processing queue (again, only if they're compatible).

3.5 Different Region Models for Segmentation

We have encapsulated the processing steps (2) initialization, (5) point compatibility, (6) model update, and (7) neighbor compatibility in exchangeable region models allowing to configure and control the behavior of the segmentation.

We have implemented several region models for approximate plane segmentation using local surface normals, as well as for segmenting regions of local surface continuity as a pre-segmentation for further processing. Both are detailed in the following. In addition, we added models that reproduce the behavior of segmentation algorithms found in the literature.

Approximate Plane Segmentation. For plane segmentation, we initialize the centroid and the normal of the region model using the seed point and its normal (instead of computing them using an initial set of points as in the approach of Poppinga et al. [13]). In order to determine the compatibility of a point $\mathbf{p}_i$ to the model, we simply check the angle between its normal $\mathbf{n_i}$ and the normal in the plane model, as well as $\mathbf{p}_i$'s distance to the plane. For updating the model w.r.t. $\mathbf{p}_i$, we incrementally update the plane's centroid, but instead of incrementally updating a covariance matrix to derive a plane normal from it, we incrementally update its centroid in normal space. That is, by pre-computing the surface normals on the mesh neighborhood, and approximating the plane normal by averaging over point normals, we can reduce the number of computations considerably.

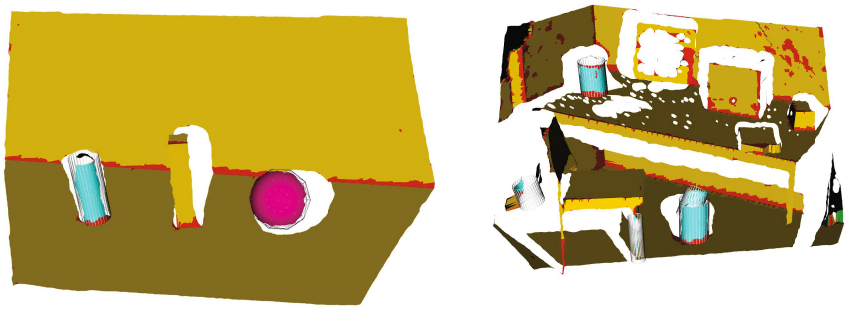
In order to obtain full polygonalizations as the one shown in Fig. 1(d), we 1) compute the convex or concave hulls (using alpha shapes) for all planar patches, and, if a triangulation is required, 2) decompose them again using ear clipping.

Extracting Locally Smooth Surfaces. For detecting geometric primitives, we need a rough pre-segmentation of the scene. This can easily be accomplished within the

neighbor compatibility check of the models by, e.g., examining changes in the local surface curvature or comparing a point's surface normal with either the region's mean surface normal or the surface normal of the seed point. A typical result of applying a segmentation using the latter model is shown in Fig. 3(d). Points on the same physical (locally smooth) surface end up in the same segment.

3.6 Detecting Geometric Primitives

For every locally smooth segment, we try to find the geometric primitive that best explains the underlying point set. Whereas we can directly compute a least squares plane fit to all the points in a segment, we use RANSAC to find the best sphere and cylinder model. Here, the computational efficiency of our approach comes from applying RANSAC only if the computed planar model does not fully explain the segment. Typical results of applying the rough pre-segmentation and primitive detection are shown in Fig. 4.



(a) Example 1 (b) Example 2

Fig. 4. Examples of detecting planes (yellow), cylinders (cyan) and spheres (magenta). Points and polygons belonging to multiple segments are colored red. All points are projected onto the found models.

Since we still stick to the RANSAC-based primitive detection for spheres and cylinders, we focus the experimental evaluation to plane segments directly obtainable from region segmentation and extracted using our approximate model.

4 Experiments and Results

We evaluate the correctness and efficiency of our approach using two publicly available data sets for which ground truth plane segmentations are available: the SegComp data set from Hoover et al. [2], and the recently published Kinect data set by Oehler et al. [17]. For the evaluation, we follow (and refer to) the scheme of [2]. Measured runtimes of the individual processing steps show that our approach can process range images at high frame rates—at a resolution of 160×120 in real-time with 30 Hz (see

Table 1). Compared the approach of Oehler et al. [17], we obtain better segmentation results (see Table 2) while being a faster by a factor of 4 (they need >100 ms for 160×120, and > 2 s for 640×480).

4.1 Estimating a Simple Isotropic Noise Model

For all the aforementioned region models, parameters like, for instance, the distance to the model and the deviation between surface normals play an important role. Whereas the latter can be neglected after applying the bilateral filter, the distance to the model is a parameter that is crucial for the quality of the segmentation. It resembles the amount of noise hindering a measurement to lie on the ideal model.

In order to obtain a rough estimate of the amount of noise at a given point, we use a simple isotropic noise model. As suggested in [18], we assume Gaussian noise $\mathcal{N}(0, \sigma^2)$ and use a simple quadratic polynomial of distance to determine σ, since noise in range sensors usually increases quadratically with the measured distance. Since the primary sensor used in our work is a Microsoft Kinect camera, we have computed a simple error model for this sensor. In 10 different scenes (ranging from scenes with only close range measurements to views of wide open space), we have collected 100 range images each. For each of the locations, we compute the mean and standard deviation per pixel and perform a least squares fit to find appropriate coefficients for the quadratic model; resulting in:

$$\sigma(d) = 0.00263d^2 - 0.00518d + 0.00752. \tag{5}$$

For the range images in the SegComp data set, that are slightly more affected by noise, we simply set $2\sigma(d)$ as the maximally allowed deviation of a point to its region model.

4.2 Plane Segmentation Using the SegComp Data Sets

Typical results of applying the presented approximate plane segmentation on range images and organized point clouds can be seen in Figures 5 and 6. Considering our goal of obtaining a fast decomposition into dominant planes and other objects of interest, the obtained results are more than satisfying. Moreover, as can be seen in the detailed comparison in Table 2, the proposed method does not considerably rank behind state-of-the-art range image segmentation approaches, while providing very efficient means to compute—within milliseconds—rough scene segmentations.

On the ABW data set [2], our approximate plane segmentation approach tends to over-segment the range image. This is caused by a special characteristic of the used camera that is not explicitly handled here resulting in inconsistent normal orientations. Besides over-segmented planar patches, our approach correctly detects 80% of the planes.

In the PERCEPTRON data set, no special sensor characteristics cause errors and our approach yields similar results as the work by Gotardo et al. [6]. Referring to Table 2, over-segmentations are rare for our approach. Instead, a considerable amount of ground truth planes is not perceived. These planes are formed by only a small number of points and are neglected here due a minimum region cardinality that we use to eliminate very small planar patches.

We also evaluated our approach using the recently published Kinect data set by Oehler et al. [17]. This data set comprises two different sets of point clouds with corresponding ground truth segmentations – one for plane segmentation and one for cylinder segmentation. Naturally, we obtain a larger amount of over-segmentations here, since larger cylinders in the planes data set are labeled as noise while unconnected regions

Table 1. Measured runtimes* for the individual processing steps

Resolution	160×120	320×240	640×480
Mesh Construction	11 ms	45 ms	188 ms
Computing Normals	2 ms	7 ms	33 ms
Smoothing/Filtering	10 ms	38 ms	155 ms
(Plane) Segmentation	6 ms	30 ms	126 ms
Overall Frequency	≈ 35 Hz	≈ 8 Hz	≈ 2 Hz

*Measured over 1000 runs on an Intel Core 2 DUO 2.26 GHz (no parallelization), using approximate plane segmentation (Sec. 3.5) of a mesh constructed using adaptive triangulation (Sec. 3.1).

(a) 10 / 14, 4 misses (b) 18 / 24, 6 misses (c) 10 / 13, 2 over, 1 miss (d) 12 / 13, 1 over (e) 22 / 22

Fig. 5. Plane segmentation using the SegComp PERCEPTRON training data set (segments randomly colored). In total, 109 of 126 planes were correctly segmented (approx. 86.5%). Not correctly found are very small plane segments, e.g., the inner parts of the objects in (a) and (b). In addition, some planes are oversegmented due to noise, e.g., the support plane in (c). The estimated plane normals deviate from ground truth by roughly $(2.5 \pm 1.6)^\circ$.

(a) 10 / 11 (b) 9 / 10 (c) 9 / 9 (d) 5 / 8, 3 over (e) 5 / 9, 2 over, 2 miss

Fig. 6. Approximate plane segmentation using (a-c) the SegComp ABW data set [2], and (d-e) the data set by Oehler et al. [17]. ABW: On average, 12.2 planes out of 15.2 planes are correctly segmented, while roughly two planes per image are oversegmented. Kinect: for the data set from [17] over-segmentations are primarily caused by detecting planes on the surface of cylinders or in small regions not marked as beloning to planes in ground truth. Overall, roughly 58 % of planes are correctly segmented (80% pixel overlap), while Oehler et al. only achieve 54.9 % (at 51 % pixel overalp, see [17]).

Table 2. Detailed benchmarking results on the SegComp data sets for plane segmentation

Approach	correctly detected	orientation deviation	over-segmented	under-segmented	missed / not detected	noise / non-existent
SegComp ABW data set (30 test images) from [2], (80 % pixel overlap as in [6])						
USF[6]	12.7 (83.5%)	1.6°	0.2	0.1	2.1	1.2
WSU [6]	9.7 (63.8%)	1.6°	0.5	0.2	4.5	2.2
UB [6]	12.8 (84.2%)	1.3°	0.5	0.1	1.7	2.1
UE [6]	13.4 (88.1%)	1.6°	0.4	0.2	1.1	0.8
OU [6]	9.8 (64.4%)	–	0.2	0.4	4.4	3.2
PPU [6]	6.8 (44.7%)	–	0.1	2.1	3.4	2.0
UA [6]	4.9 (32.2%)	–	0.3	2.2	3.6	3.2
UFPR [6]	13.0 (85.5%)	1.5°	0.5	0.1	1.6	1.4
Oehler et al. [17]	11.1 (73.0%)	1.4°	0.2	0.7	2.2	0.8
Ours	12.2 (80.1%)	1.9°	1.8	0.1	0.9	1.3
SegComp PERCEPTRON data set (30 test images) from [2], (80 % pixel overlap as in [6])						
USF [6]	8.9 (60.9%)	2.7	0.4	0.0	5.3	3.6
WSU [6]	5.9 (40.4%)	3.3	0.5	0.6	6.7	4.8
UB [6]	9.6 (65.7%)	3.1	0.6	0.1	4.2	2.8
UE [6]	10.0 (68.4%)	2.6	0.2	0.3	3.8	2.1
UFPR [6]	11.0 (75.3%)	2.5	0.3	0.1	3.0	2.5
Oehler et al. [17]	7.4 (50.1%)	5.2	0.3	0.4	6.2	3.9
Ours	11.0 (75.3%)	2.6	0.4	0.2	2.7	0.3

belonging to a single plane are labeled as one segment. However, visually inspecting the segmentation results reveals that all dominant planes are reliably segmented.

5 Conclusions and Future Work

We have presented a simple, yet efficient approach to segmenting range images and organized point clouds. Using an approximate polygonal reconstruction directly deduced from the image-like structure, we are able to efficiently compute point features such as local surface normals, and to smooth the measured data using a bilateral filter.

Experimental evaluation has shown that our approach does not considerably rank behind state-of-the-art range image segmentation techniques. However, regarding its computational complexity, it exploits several simplifications and approximations finally making the overall segmentation (and primitive detection) algorithm run within milliseconds on point clouds acquired by typical RGB-D cameras.

It remains a matter of future work to exploit the extracted information about segmented planar patches (and geometric primitives) in further processing steps like registration.

Implementations of all components presented in this paper are (or are going to be) publicly available within the open source Point Cloud Library PCL[1].

[1] The latest stable release of the Point Cloud Library PCL is available at `http://pointclouds.org`

References

1. Rusu, R.B., Blodow, N., Marton, Z.C., Beetz, M.: Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in human environments. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, MO, USA, pp. 1–6 (2009)
2. Hoover, A., Jean-Baptiste, G., Jiang, X., Flynn, P.J., Bunke, H., Goldgof, D.B., Bowyer, K., Eggert, D.W., Fitzgibbon, A., Fisher, R.B.: An experimental comparison of range image segmentation algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 18, 673–689 (1996)
3. Vosselman, G., Gorte, B.G.H., Sithole, G., Rabbani, T.: Recognising structure in laser scanner point clouds. Information Sciences 46(8), 1–6 (2004)
4. Lee, K.-M., Meer, P., Park, R.-H.: Robust adaptive segmentation of range images. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 200–205 (1998)
5. Silva, L., Bellon, O., Gotardo, P.: A global-to-local approach for robust range image segmentation. In: Proc. of the Int. Conference on Image Processing (ICIP), Rochester, NY, USA, pp. 773–776 (2002)
6. Gotardo, P., Bellon, O., Silva, L.: Range image segmentation by surface extraction using an improved robust estimator. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Madison, WI, USA, pp. 33–38 (2003)
7. Schnabel, R., Wahl, R., Klein, R.: Efficient RANSAC for point-cloud shape detection. Computer Graphics Forum 26(2), 214–226 (2007)
8. Holz, D., Schnabel, R., Droeschel, D., Stückler, J., Behnke, S.: Towards semantic scene analysis with time-of-flight cameras. In: Proc. of the 14th RoboCup International Symposium, Singapore, (2010)
9. Holz, D., Holzer, S., Rusu, R.B., Behnke, S.: Real-Time Plane Segmentation Using RGB-D Cameras. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 306–317. Springer, Heidelberg (2012)
10. Stückler, J., Steffens, R., Holz, D., Behnke, S.: Real-time 3D perception and efficient grasp planning for everyday manipulation tasks. In: Proc. of the European Conference on Mobile Robots (ECMR), Örebro, Sweden, pp. 177–182 (2011)
11. Borrmann, D., Elseberg, J., Lingemann, K., Nuechter, A.: The 3D Hough transform for plane detection in point clouds: A review and a new accumulator design. 3D Research 2, 1–13 (2011)
12. Hähnel, D., Burgard, W., Thrun, S.: Learning compact 3D models of indoor and outdoor environments with a mobile robot. Robotics and Autonomous Systems 44(1), 15–27 (2003)
13. Poppinga, J., Vaskevicius, N., Birk, A., Pathak, K.: Fast plane detection and polygonalization in noisy 3D range images. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, pp. 3378–3383 (2008)
14. Harati, A., Gächter, S., Siegwart, R.: Fast range image segmentation for indoor 3D-SLAM. In: Proc. of the IFAC Symposium on Intelligent Autonomous Vehicles (IAV), Toulouse, France (2006)
15. Rabbani, T., van den Heuvel, F., Vosselman, G.: Segmentation of point clouds using smoothness constraint. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 36, 248–253 (2006)
16. Cupec, R., Nyarko, E.K., Filko, D.: Fast 2.5D mesh segmentation to approximately convex surfaces. In: Proc. of the European Conference on Mobile Robots (ECMR), Örebro, Sweden, pp. 49–54 (2011)

17. Oehler, B., Stueckler, J., Welle, J., Schulz, D., Behnke, S.: Efficient Multi-resolution Plane Segmentation of 3D Point Clouds. In: Jeschke, S., Liu, H., Schilberg, D. (eds.) ICIRA 2011, Part II. LNCS, vol. 7102, pp. 145–156. Springer, Heidelberg (2011)
18. Anderson, D., Herman, H., Kelly, A.: Experimental characterization of commercial flash ladar devices. In: International Conference of Sensing and Technology, Palmerston North, New Zealand (2005)