Accepted for Robotics and Autonomous Systems, to appear 2014.

Approximate Triangulation and Region Growing for Efficient Segmentation and Smoothing of Range Images

Dirk Holz and Sven Behnke

University of Bonn, Autonomous Intelligent Systems Group, Friedrich-Ebert-Allee 144, 53113 Bonn, Germany

Abstract

Decomposing sensory measurements into coherent parts is a fundamental prerequisite for scene understanding that is required for solving complex tasks, e.g., in the field of mobile manipulation. In this article, we describe methods for efficient segmentation of range images and organized point clouds. In order to achieve real-time performance in complex environments, we focus our approach on simple but robust solutions. We present a fast approach to surface reconstruction in range images and organized point clouds 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. A comparative evaluation using publicly available data sets shows that our approach achieves state-of-the-art performance while being significantly faster than other methods.

Keywords: Scene understanding, range image segmentation, approximate triangulation, multilateral smoothing

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 for this is the decomposition of sensory measurements into homogeneous parts that are relevant for the tasks of the robot. For mobile manipulation in complex environments, for example, the perception of objects and their surroundings is a key prerequisite. A common approach [29] in 3D object (and environment) perception is to exploit typical

☆This research has been partially funded by the FP7 ICT-2007.2.2 project ECHORD (grant agreement 231143) experiment ActReMa and the FP7 ICT-2013.2.2 project STAMINA (grant agreement 610917).

Email address: `{holz, behnke}@ais.uni-bonn.de` (Dirk Holz and Sven Behnke)

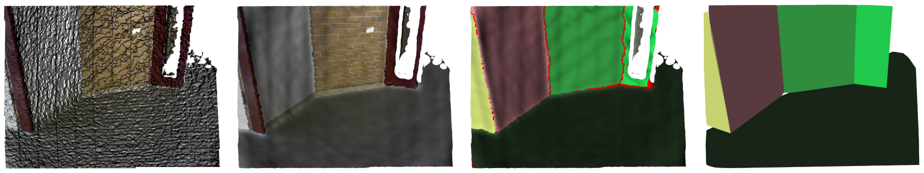

(a) Approx. mesh (b) Smoothed mesh (c) Segmentation (d) Polygonalization

Figure 1: Example of surface reconstruction and (plane) segmentation on a RGB-D point cloud: (a) initial approximate triangulation; (b) smooth mesh after multilateral filtering; (c) result of segmenting planes in the mesh (red triangles are assigned to multiple planes and lie along borders); (d) polygonalization as a collection of alpha shapes.

characteristics of man-made environments and to apply the following processing pipeline:

1.) detect horizontal support planes,

2.) extract and cluster points on top of these planes, and

3.) perform further processing in the found clusters, e.g., recognizing, classifying or tracking objects.

One of the fundamental challenges in this pipeline 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 using only a single CPU. 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. Through the use of easily exchangeable components, our generalized region growing approach allows for different region models (e.g., planes) to be segmented in the data. We present models for segmenting planes, regions of local surface continuity, and simple geometric primitives at high frame rates (see Figure 1).

We further use the same mesh neighborhoods to efficiently compute local surface normals and curvature estimates, as well as to smooth both the 3D measurements and the computed normals using a multilateral filter.

This paper is an extended version of our previous work [12]. It is organized as follows: After a discussion of related work on range image and 3D plane segmentation methods in Section 2, we give an overview on our approach in Section 3. Our methods for approximate surface reconstruction, efficient computation of local surface normals and curvature, and filtering of the constructed mesh are presented in Section 4. In Section 5, we describe our generalized region growing algorithm as well as different models for plane segmentation and the detection of other geometric primitives. We investigate different camera noise models that assist both initial mesh construction and segmentation in Section 6. We evaluate efficiency and robustness of our approach on multiple publicly available data sets and summarize the results in Section 7. We show that our approach achieves state-of-the-art performance while being faster than other methods.

2. Related Work

Research on computer and robot vision has yielded a wide variety of approaches to range image segmentation—and plane segmentation in particular. Hoover et al. [16] compiled a survey and performed an evaluation of early work. For an overview on more recent work, we refer to the survey of Vosselman et al. [36]. In principle, four different types of approaches can be distinguished according to the underlying working principle: methods using variants of the random sample consensus (RANSAC) algorithm [6], 3D Hough transforms, scan line grouping, and region growing.

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. [18] sequentially find a model using RANSAC, remove inliers from the original data set, and continue the segmentation with the residual points. Silva et al. [31] first identify connected regions and apply RANSAC region-wise. Gotardo et al. [8] compute an edge map for pre-segmentation and fit model parameters using a robust estimator based on the M-estimator sample consensus (MSAC) by Torr and Zisserman [34].

Another efficient solution to the problem of segmenting even unorganized point clouds and detecting simple geometric primitives such as planes, cylinders, and spheres has been proposed by Schnabel et al. [30]. They decompose unorganized point clouds using octree subdivision and apply RANSAC only to subsets of the original point cloud.

In previous work [14], we adapted the perception scheme from Section 1 as well as the techniques from [29] and [30], 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. [30] only to already extracted and segmented points above that plane. In [13], we further sped 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 RGB-D camera and has been used for real-time object tracking and grasp planning [33].

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 of Hough-based segmentation approaches, we refer to the works of Vosselman et al. [36] and Borrmann et al. [3].

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. For example, in a shelf with vertical separators, boards on the same level are merged, although they do not physically lie on the same surface. In addition, Hough-based segmentation may suffer from discretization effects.

In [13], 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 (in contrast to scan line grouping and the region growing-based approaches following).

2.3. Scan Line Grouping

In the context of segmenting 3D laser range scans, another popular approach is *scan line grouping*. It aims for computational efficiency by first detecting lines in planar cuts (and 2D range scans forming a 3D scan), and by merging neighboring line segments to regions in a second step. The basic idea behind this principle is the observation that planes in a 3D scan form straight lines in two-dimensional scan lines and that points on the same line segment belong to the same 3D plane [17]. Jiang and Bunke [17] store detected line segments in a link-based data structure that eases region growing for extracting planar patches from detected lines. Nüchter et al. [24] follow a similar approach but use RANSAC and ICP for plane detection, and label detected planes as belonging to floor, ceiling and walls. Gutmann et al. [9] improve various aspects of the original formulation by computing and analyzing statistics about points on a scan line for splitting line segments and growing regions. An et al. [1] first cluster the scan lines and only consider the end points of line segments to speed up line and plane detection. Recently, Georgiev et al. [7] applied scan line grouping on data acquired from RGB-D cameras. We include the approach of Georgiev et al. in our experimental evaluation.

2.4. Segmentation using Region Growing

The idea of region growing-based segmentation is to exploit the image-like data structure of organized point clouds. Hähnel et al. [10] 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. [27] 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 distances to the currently grown plane lie below a threshold. The centroid and covariance matrix for estimating the plane's parameters are thereby updated incrementally.

Here, we follow a similar approach for segmenting planes. 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 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. [8] for RANSAC-based segmentation, Harati et al. [11], first compute an edge map to find connected regions of local surface continuity. Rabbani et al. [28] 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. [5] 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 multilateral filter. Our framework allows for using different types of models for region growing, including the approaches of Rabbani et al. [28], Cupec et al. [5] and Poppinga et al. [27], as well as our fast approximation (see Section 5.2).

3. Approach

The overall processing pipeline of our approach is composed of the following components:

1. Deducing an approximate mesh from the image neighborhoods.
2. Using the mesh neighborhoods to compute approximate local surface normals and curvature estimates.
3. Multilateral filtering to smooth both points and normals.
4. Segmentation based on region growing (using different region models depending on the desired segmentation).

As an optional last step, we replace found segments by the geometric primitives best fitting the contained points in order to obtain a compact representation. Planes are replaced by polygons, e.g., the convex hull and the model parameters for the plane. An overview on our system is presented in Figure 2.

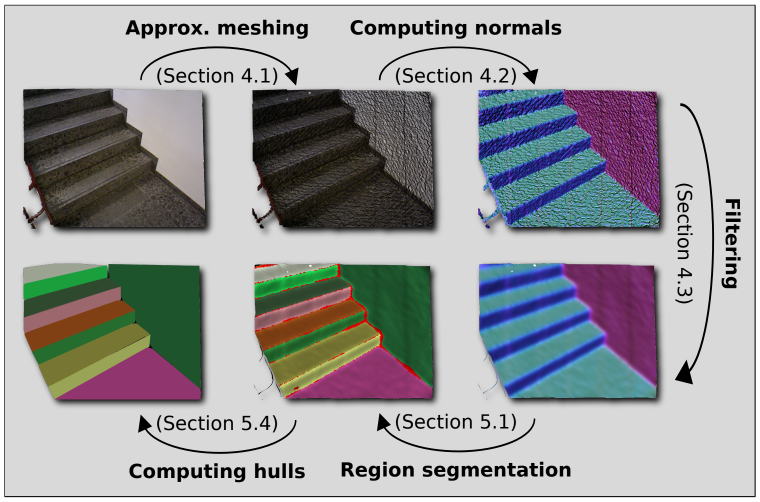


Figure 2: Processing pipeline (following the black arrows from top left): for an input organized point cloud or range image, we first deduce an approximate triangle or quad mesh. We efficiently compute local surface normals and curvature directly on the mesh and apply a multilateral filter to smooth both the points and their normals. The smoothed mesh is then segmented into planar regions. In a last (optional) processing step, detected planes are replaced by polygons.

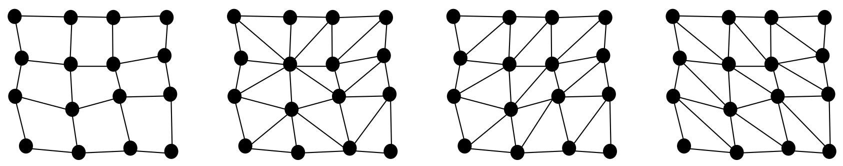
Compared to related work, our approach is particularly efficient since local point neighborhoods as well as distances and changes in surface orientations between neighboring points are not only computed efficiently using the approximate mesh but also cached in the mesh structure for further processing. All components described in the above list use the same cached neighborhoods.

4. Fast Approximate Surface Reconstruction

4.1. 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 following algorithms could easily be applied on local index neighborhoods in range images. However, an approximate mesh allows for 1) the application of a wide variety of sophisticated algorithms for processing meshes, and 2) the storage of edge weights for caching point attributes or relations between points, e.g., differences in local surface normal orientations or the difference vectors for integral image-based normal computation as in our previous work [13].

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



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

Figure 3: 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.

- $R(u, v)$ 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.

  and

- 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 are exploiting. 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 efficiently done by examining the difference vectors between $\mathbf{p}_i$ and its three neighbors. If one of the difference vectors falls into a common line of sight with the viewpoint from where the measurements were taken (the focal point $\mathbf{f} = \mathbf{0}$), then one of the underlying surfaces occludes the other. The condition for having an occluded edge between point $\mathbf{p}_i$ and its neighbor $\mathbf{p}_j$ can be formulated as

$$\begin{aligned} \mathit{valid} &= \left(|\cos\theta_{i,j}| \leq \cos\epsilon_\theta\right) \wedge \left(d_{i,j} \leq \epsilon_d^2\right), && (1) \\ \text{with} \quad \theta_{i,j} &= \frac{(\mathbf{p}_i - \mathbf{f}) \cdot (\mathbf{p}_i - \mathbf{p}_j)}{\|\mathbf{p}_i - \mathbf{f}\| \, \|\mathbf{p}_i - \mathbf{p}_j\|}, && (2) \\ \text{and} \quad d &= \|\mathbf{p}_i - \mathbf{p}_j\|^2, && (3) \end{aligned}$$

where ϵ_θ and ϵ_d denote maximum angular and length tolerances, respectively. In the same way, we can check for jump edges [21], i.e., erroneous measurements often induced by range sensors in the vicinity of occlusions and depth discontinuities.

If all checks pass, $R(u, v)$ and its neighbors are used to extend the so far built mesh. Otherwise, holes arise. Referring to Fig. 3, 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 $R(u+1, v)$.

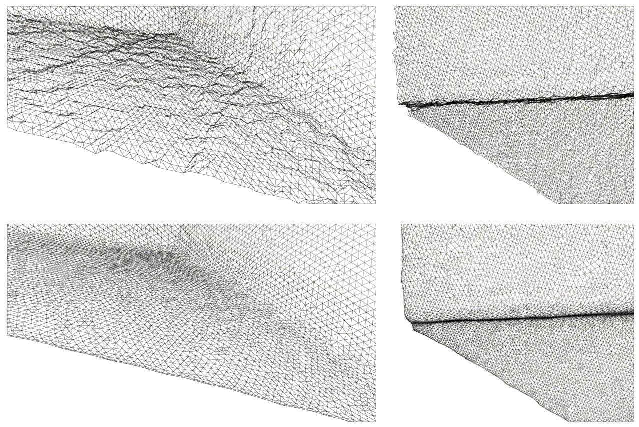


Figure 4: Two views on an adaptive triangular mesh (a) before and (b) after filtering. The surface is considerably smoothed while preserving edges and corners.

2. Fixed left cut 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. Compared to the fixed triangulations, it achieves a higher accuracy in the vicinity of edges.

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. Example triangulations are shown in Figures 1 and 2.

4.2. 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 N_T 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:

$$\mathbf{n}_i = \frac{\sum_{j=0}^{N_T}(\mathbf{p}_{j,a} - \mathbf{p}_{j,b}) \times (\mathbf{p}_{j,a} - \mathbf{p}_{j,c})}{\|\sum_{j=0}^{N_T}(\mathbf{p}_{j,a} - \mathbf{p}_{j,b}) \times (\mathbf{p}_{j,a} - \mathbf{p}_{j,c})\|}, \tag{4}$$

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 products, and add them to the normals of the involved points. Finally, we normalize all point normals at once. An example of computed local surface normals (color coded) can be seen in Figure 2.

Since we compute local surface normals on the mesh deduced from the range image and not on the range image itself (as in the case of integral image-based normal estimation [13, 15]) we get proper normal estimates even in the vicinity of depth discontinuities whereas image-based methods tend to smooth over edges and depth discontinuities. We approximate the local surface curvature of a point by the standard deviation of the normal directions of its neighbors [19].

4.3. Multilateral Filtering

Naturally, sensor measurements are affected by noise. Since this noise may hinder further processing, e.g. segmentation, we apply a filter for smoothing both the points and their normals while preserving edges in the sensed geometric structures. The formulation of our filter is motivated by the concept of multilateral filtering [4] and measures the similarity of points w.r.t. their position, surface orientation, and appearance. As for the other components in our pipeline, we directly extract the neighborhood of a point from the mesh instead of searching for nearest neighbors. 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:

$$\mathbf{p}_i = \sum_{j \in N_i} w_{ij} \mathbf{p}_j / \sum_{j \in N_i} w_{ij}, \quad \text{and} \quad \mathbf{n}_i = \sum_{j \in N_i} w_{ij} \mathbf{n}_j / \sum_{j \in N_i} w_{ij}, \tag{5}$$

$$\text{with} \quad w_{ij} = \underbrace{e^{\alpha \|\mathbf{p}_i - \mathbf{p}_j\|}}_{\text{distance term}} \underbrace{e^{\beta \|\mathbf{n}_i - \mathbf{n}_j\|_1}}_{\text{normal term}} \underbrace{e^{\gamma (I_i - I_j)/c_I}}_{\text{intensity term}}, \tag{6}$$

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 $[0, 1]$. Weights α, β, and γ can be used to adjust the behavior of the filter. Equally weighting distance and surface normal deviation term already achieves considerable smoothing while preserving edges and corners. Depending on the desired smoothing level, we can extend a point's neighborhood to include the neighbors of neighbors and ring neighborhoods farther away from the point. An example of filtering an input mesh (weights $\alpha = 1$, $\beta = 1$, $\gamma = 0$) can be seen in Figure 4.

5. Region Models and Segmentation

For being able to efficiently segment 3D data of various types, e.g., range images as well as both organized and unorganized point clouds, we have implemented a generalized segmentation framework. It allows the involved components such as the underlying region model (e.g., planes or non-planar locally smooth surfaces), the module for estimating the sensor noise, and the method for obtaining the local neighborhoods of query points (and the corresponding distances and surface normal deviations) to be easily switched. Whereas for organized data, we only require a look-up in the initially constructed and smoothed mesh, for unorganized data we need to either apply the greedy projection-based surface reconstruction algorithm by Marton et al. [20] or search for the neighbors for all points using fast approximate neighbor search [22] and cache the neighbors for later use.

5.1. 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 literature. Given is a set of seed points (and a priority queue of seeds) or simply the array of all points.

Outer loop, until all points are processed:

1) 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) Take the next point from the processing queue (and go back to 1) if it is empty),
   5) check the compatibility of the point with the region model, and
   6) add it in case of compatibility. If necessary, update the region model w.r.t. the new point.
   7) Add the neighbors of the point to the processing queue if they pass a neighbor compatibility check.

5.2. 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 the behavior of the segmentation to be configured and controlled. Note that we distinguish two types of compatibility checks—one for points determining whether or not they belong to the currently grown region and one for a point's neighbors determining whether or not they are added to the processing queue at all.

We have implemented several region models for plane segmentation—a probabilistic incremental formulation based on [27] and an approximate variant using local surface normals—as well as for segmenting regions of local surface continuity as a pre-segmentation for further processing. In addition, we added models that reproduce the behavior of other segmentation algorithms found in the literature, amongst others, the approaches of Rabbani et al. [28] and Cupec et al. [5].

5.3. Probabilistic Plane Segmentation

In order to reliably detect planes even in noisy data, we use a probabilistic region model that is based on the incremental plane fitting algorithm of Poppinga et al. [27]. Aimed at time-of-flight cameras, their approach exploits the sequential structure of the data and uses incremental updates of both the centroid and covariance matrix of the inliers. The centroid and covariance matrix are not only used to determine the plane parameters, but also the uncertainty of the estimated plane parameters. A point is added to the probabilistic region model if the incrementally updated mean square error of the plane fit and the point's distance to the estimated plane model do not exceed a threshold. Despite the efficient incremental updates to centroid, covariance and mean square error, the probabilistic model is computationally more demanding than other region models in our framework, but achieves reliable plane detection results (Section 7).

5.4. Approximate Plane Segmentation

In order to further speed up plane segmentation while preserving reliable plane detections, we have developed an approximate variant of the probabilistic plane segmentation model. For approximate plane segmentation, we initialize the centroid and the normal of the region model using the seed point and its normal. Both can be taken directly from the smoothed approximate mesh instead of computing an initial centroid and covariance matrix using an initial set of points as in the probabilistic approach. In order to determine the compatibility of a point $\mathbf{p}_i$ with the model, we simply check the angle between its normal $\mathbf{n_i}$ and the normal of the plane model, as well as $\mathbf{p}_i$'s distance to the plane. To incorporate $\mathbf{p}_i$ in the model, 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 precomputing the surface normals on the mesh neighborhood, and approximating the plane normal by averaging over point normals, we can reduce the number of required computations considerably. Note, for assessing the uncertainty in the determined plane parameters (e.g., for the registration of planar segments as in [26]), we can compute the necessary Hessian and covariance matrices after region growing using the final sets of inliers.

In order to obtain full polygonalizations such as the one shown in Figure 2, we compute the convex or concave hulls (using alpha shapes) for all planar patches. In case a triangulation is required for further processing, we decompose resulting polygons again using ear clipping (which is fast and produces satisfactory results in most cases). This post-processing step allows for the creation of highly efficient scene representations—thousands of triangles or quads are replaced by a fraction as many polygons, while still representing all dominant planes.

5.5. Extracting Locally Smooth Surfaces and Detecting Geometric Primitives

For detecting geometric primitives, we need a rough pre-segmentation of the scene. This can easily be accomplished within the neighbor compatibility

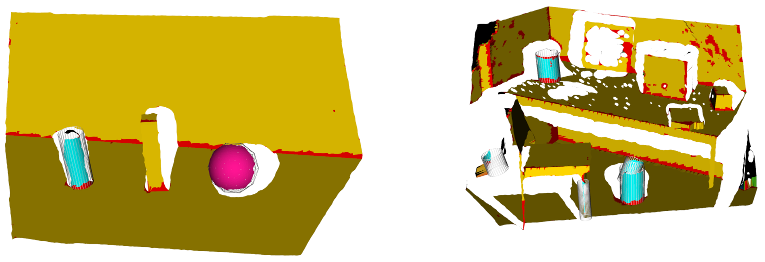


Figure 5: Examples of detecting planes (yellow), cylinders (cyan), and spheres (magenta). Points and polygons belonging to multiple segments are colored red. All points are projected onto the found models.

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 that uses the latter model is shown in Fig. 5. Points on the same physical (locally smooth) surface end up in the same segment.

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. 5.

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.

6. Camera Noise Models

The key to both the initial construction of the approximate mesh and its segmentation is to know whether or not neighboring points in the range image are close to each other and lie on the same physical surface.

For all of the aforementioned region models, parameters such as, for instance, the distance to the model and the deviation between surface normals play an important role. Whereas the latter can (almost) be neglected after applying the multilateral 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 from lying 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 [2], we assume Gaussian noise $\mathcal{N}(0, \sigma^2)$ and use a simple quadratic polynomial as a function 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 RGB-D 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:

$$\sigma_{\mathrm{FIT}}(z) = 0.00263z^2 - 0.00519z + 0.00755. \tag{7}$$

This simple model considers only the expected measurement noise but already provides a good estimate (see Section 7.2) although it neglects both the characteristic camera errors induced by the quantization of measurements, and the angle to the surface that measurements are acquired on. Measurements taken at extreme angles (e.g., on walls while traversing a corridor) are considerably more affected by noise. Nguyen et al. [23] measure both lateral and axial noise distributions as a function of both distance and angle to the sensed surfaces. They report an (almost) angle independent error when neglecting spurious data measured under extreme angles (smaller than 10° or larger than 60°). For this regular case their approximation is:

$$\sigma_{\mathrm{Nguyen}}(z) = 0.0019\,(z - 0.4)^2 + 0.0012. \tag{8}$$

Holzer et al. [15] neglect the angle-dependent error and provide a noise model solely based on the quantization effect induced by the measurement principle:

$$\sigma_{\mathrm{Holzer}}(z) = 0.0028z^2. \tag{9}$$

Smisek et al. [32] conduct a similar set of experiments and assess the quantization effect in a Microsoft Kinect camera as being

$$\sigma_{\mathrm{Smisek}}(z) = 0.00273z^2 + 0.00074z - 0.00058. \tag{10}$$

We have conducted a set of experiments (see Section 7.2) to evaluate the influence of the noise model used for both approximate meshing and region segmentation. Although the final segmentation performance does not considerably deviate for the different noise models, the best results can be achieved with a combination of the quantization-based models (Holzer et al. [15] and Smisek et al. [32]) and the fitting-based models (Nguyen et al. [23] and our σ_{FIT}):

$$\sigma_{\mathrm{Holz}}(z) = \begin{cases} \sigma_{\mathrm{FIT}}(z) & \text{if } z \leq 0.85\,\mathrm{m} \quad \text{where } \sigma_{\mathrm{FIT}}(z) = \sigma_{\mathrm{Holzer}}(z) \\ \sigma_{\mathrm{Holzer}}(z) & \text{if } z > 0.85\,\mathrm{m}. \end{cases} \tag{11}$$

The different natures of the two model types are reflected by the two curve clusters in Figure 6.

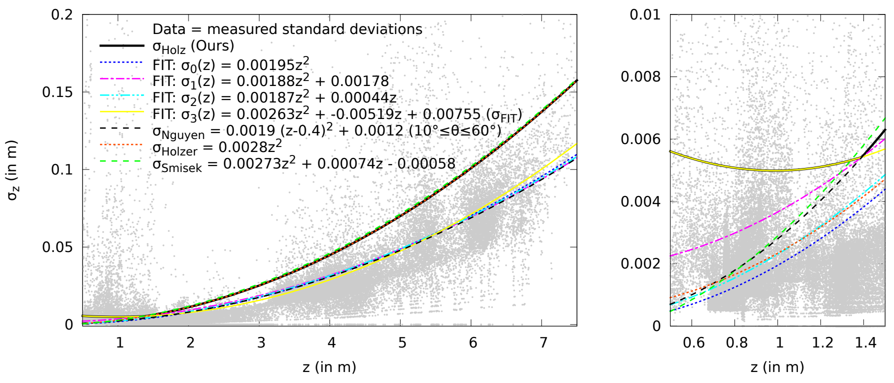


Figure 6: Isotropic noise models as estimated for Microsoft Kinect cameras (left). The detail view (right) shows that most models underestimate the noise in close ranges, compared to our fitted σ_3 model.

7. Experiments and Results

In order to assess the performance of our approach and the influence of the individual components, we perform a set of experiments. We evaluate the correctness and efficiency of our approach using two publicly available data sets for which ground truth plane (and cylinder) segmentations are available: the SegComp data set[1] by Hoover et al. [16], and the recently published Kinect data set[2] by Oehler et al. [25]. The latter follows the same file formats and conventions as the SegComp data sets. From the SegComp data set, we have used the *ABW* and the *PERCEPTRON* parts. For the evaluation, we follow (and refer to) the scheme of Hoover et al. [16]. The two SegComp parts consist of 10 training and 30 test images. For all images ground truth segmentations and plane parameters are available. The evaluation tool averages over all test images correctly segmented planes, oversegmented planes, undersegmented planes and false detections (labeled as "noise"). In addition, the tool assesses the average deviation of the estimated plane orientations from ground truth. The Kinect data set by Oehler et al. [25] consists of two parts—one for planes and one for cylinders—with 30 organized RGB-D point clouds each. Training and test data is not distinguished and the evaluation includes all point clouds. Ground truth plane orientations are not available for the Kinect data set. However, Oehler et al. provide a modified evaluation tool[2] that neglects plane orientations. The following comparative evaluations include segmentation results gathered by Gotardo et al. [8] and Oehler et al. [25], achieved results using publicly available implementations of Georgiev et al. [7] and Trevor et al. [35], our previous work [13], and the plane and cylinder segmentation models used in this work.

[1]The SegComp data set is available at: `http://marathon.csee.usf.edu/seg-comp`.

[2]The data set of Oehler et al. is available at: `http://www.ais.uni-bonn.de/download/segmentation/kinect.html`.

Table 1: Measured runtimes for the individual processing steps for Kinect frames (160×120, 320×240, 640×480), and training and testing images from the SegComp *ABW* and *PERCEPTRON* data sets (512 × 512).

Resolution	160×120	320×240	512×512	640×480
Approximate meshing	6.1 ms	17.2 ms	40.4 ms	48.4 ms
Computation of normals	1.5 ms	3.6 ms	12.2 ms	14.1 ms
Filtering	4.5 ms	11.6 ms	27.4 ms	33.5 ms
(Plane) segmentation	3.2 ms	11.3 ms	26.3 ms	32.9 ms
Overall frequency	$\approx$65 Hz	$\approx$23 Hz	$\approx$10 Hz	$\approx$7.7 Hz

*Runtimes measured over 10 000 runs on an Intel Core i7 CPU @ 2.7 GHz (no parallelization), approximate plane segmentation (5.4) of a mesh constructed using adaptive triangulation (4.1).

7.1. Runtime Evaluation

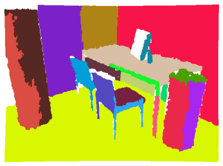
In order to assess the runtime of the overall approach and of the involved components, we have defined a baseline system consisting of the following components: adaptive triangular meshing, our noise model σ_{Holz}, the multilateral filter with weights $\alpha = 1$, $\beta = 1$, $\gamma = 0$, and the region model for approximate plane segmentation. The baseline system is applied to all training and test images of both the SegComp and the Kinect data sets. The Kinect data set has been downsampled to assess the runtimes for all resolutions offered by the camera (see Table 1). We can segment all dominant planes with roughly 7.7 Hz at VGA resolution, and more than two times faster than the camera measurement frequency at the downsampled 160 × 120 resolution.

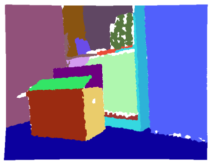
Compared to the approach of Oehler et al. [25], we obtain better segmentation results while being faster by a factor of eight (they report runtimes of >100 ms for 160×120, and > 2 s for 640×480). The approach of Trevor et al. [35] achieves roughly 10 Hz (VGA) at the cost of inferior segmentation results, just as with the approach of Georgiev et al. [7] with roughly 8 Hz (VGA). For both implementations parameters yielding the best results have been chosen. Slightly faster than the approach of Trevor et al. [35] is our clustering method from previous work [13] with roughly 15 Hz (VGA), again at the cost of segmentation performance. Using the probabilistic plane segmentation model (instead of our approximate one), the runtime for the segmentation step in Table 1 increases to approximately 95 ms leading to an overall computation frequency of roughly 5 Hz at VGA resolution.

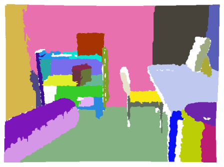
7.2. Segmenting Organized Point Clouds and the Influence of the Noise Model

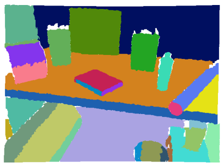
We have conducted a set of experiments to assess the influence of the applied noise model on both approximate meshing and segmentation. Using the Kinect planes data set by Oehler et al. [25], we evaluated the average plane detection results for the same baseline system as in the runtime evaluation (see Section 7.1), but with different noise models.

The data set comprises two different parts with corresponding ground truth segmentations—one for plane segmentation and one for cylinder segmentation.


(a) 5/9, 2 over, 2 miss


(b) 11/14, 2 over, 1 miss


(c) 9/13, 2 over, 2 miss


(d) 5/8, 3 over

	Approach	correctly detected	over-segmented	under-segmented	missed (not detected)	noise (nonexistent)
	Oehler et al. [25]	4.50 (36.3 %)	0.60	0.40	6.80	16.20
Noise model	σ_{FIT}	7.10 (57.2 %)	3.63	0.10	1.46	13.20
	σ_{Nguyen} [23]	7.10 (57.2 %)	3.63	0.10	1.46	13.20
	σ_{Holzer} [15]	7.20 (58.0 %)	3.67	0.10	1.33	13.00
	σ_{Smisek} [32]	7.20 (58.0 %)	3.67	0.10	1.33	13.00
	σ_{Holz}	**7.23 (58.3%)**	3.70	0.10	1.27	12.90

(e) Overall plane segmentation results for all 30 point clouds (80 % pixel overlap).

Figure 7: Plane segmentation on the Kinect planes data set by Oehler et al. [25]. Larger cylinders are segmented into multiple planes. Overall, about 58% of the 12.4 planes are correctly segmented using the baseline system (assuming 80% pixel overlap). The performance does not considerably deviate with the different noise models.

Naturally, we obtain a larger amount of oversegmentations here, since larger cylinders in the planes part are labeled as noise, while unconnected regions belonging to a single plane are labeled as one segment. However, visually inspecting the segmentation results reveals that all dominant planes are reliably segmented.

Typical plane segmentation results (for the baseline system with model σ_{Holz}) as well as the detailed segmentation results are shown in Figure 7. Although the final segmentation performance does not considerably deviate for the different noise models under consideration, visual inspection of the triangulation results showed that the quantization-based models (Holzer et al. [15] and Smisek et al. [32]) underestimate the noise in close ranges (leading to oversegmentations) while the fitting-based models (Nguyen et al. [23] and our σ_{FIT}) underestimate at larger distances (causing missed detections). In the data set this does not further affect the overall detection results and is only reflected by missing or not missing few very small planes and oversegmenting or not oversegmenting a few very close planes. Combining the two classes in the new noise model with σ_{Holz} yields the best results.

7.3. Plane Segmentation using the SegComp Data Sets

In the following, we present our plane detection results for the publicly available SegComp data sets *PERCEPTRON* and *ABW* by Hoover et al. [16]. Since our noise model σ_{Holz} has been developed for RGB-D cameras, and the Microsoft Kinect in particular, we have computed a simple noise model for the SegComp data sets by fitting a quadratic polynomial to the mean square distance of the measurements in the *PERCEPTRON* and *ABW* training images to the underlying ground truth planes. The resulting noise model can be approximated by:

$$\sigma_{\text{SegComp}}(z) = 0.0036z^2. \qquad (12)$$

We use the same noise model for both *PERCEPTRON* and *ABW*.

Typical results of applying the presented approximate plane segmentation on range images of the two data sets can be seen in Figures 8 and 9. 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 it can be seen in the detailed comparisons in the result tables of Figures 8 and 9, the proposed approximate plane segmentation method and the probabilistic model achieve state-of-the-art range image segmentation performance, while providing very efficient means to compute—within milliseconds—rough scene segmentations.

On the *ABW* data set [16], our approximate plane segmentation approach tends to oversegment 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 oversegmented planar patches, both plane detection models correctly detect more than 80% of the planes. The approach of Georgiev et al. [7] considerably undersegments planes meeting at obtuse angles, while the approach of Trevor et al. [35] misses smaller planar patches.

In the *PERCEPTRON* data set, no special sensor characteristics cause errors and our approach yields similar results as the work by Gotardo et al. [8]. Referring to the detailed result tables in Figures 8 and 9, oversegmentations are rare for our approach. Instead, a considerable number of ground truth planes are not perceived. These planes are formed by only a small number of points and are neglected here due to a minimum region cardinality $|R|= 200$ that we use to eliminate outliers and very small planar patches. The approach of Georgiev et al. [7] considerably oversegments in particular smaller planar patches. Insufficient pixel overlap causes a considerable amount of missed planes (especially the dominant support planes) in the approach of Trevor et al. [35].

Our previous approach [13] suffers from the same inconsistent normal orientations in the *ABW* data set and tends to oversegment. Furthermore, it does not consider models of the underlying noise, but uses fixed distance-independent thresholds. As a consequence, the algorithm tends to oversegment in both the *ABW* and the *PERCEPTRON* data set. Although it merges neighboring plane clusters to compensate for discretization effects in the histograms, not all oversegmentations are resolved. In addition, since the approach does not consider the neighborhood of a pixel, larger normal deviations in noisy regions can lead to individual pixels not belonging to the same planar segment as its neighbors. Since the evaluation considers an 80% pixel overlap with the ground truth segmentations, the pixels left out cause some planes to be missed. Overall, our previous approach [13] achieves similar performance as the algorithms of Georgiev et al. [7] and Trevor et al. [35], but ranks behind state-of-the-art segmentation performance. However, it was the fastest in our experiments (15 Hz at VGA), roughly 20 % to 25 % faster than the proposed approach.

7.4. Cylinder Segmentation

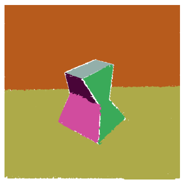
In order to assess the reliability of our approach to detect simple geometric primitives, we have used the Kinect cylinder data set by Oehler et al. [25]. As described in Section 5.5, we first segment the recorded RGB-D point clouds into regions of local surface continuity and then fit plane, cylinder, and sphere models to each region and select the model best supporting the underlying point set. Both detected planes and spheres (no spheres were detected in the data set) are marked as belonging to the background, since the data set contains only ground truth pixels for cylinders whereas all other pixels are labeled as noise. Typical results of cylinder detection as well as the overall segmentation performance on the Kinect cylinder data set are shown in Figure 10.

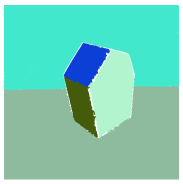
All larger cylinders in the data set are reliably detected. Cylinders being sensed from above, i.e., where both outer and inner part are visible, reveal a systematic effect of our approach. In the segmentation of regions with local surface continuity, the two parts are split since the outer part occludes the inner part and is unconnected in the mesh, and the small connected regions to the sides show discontinuities in the surface normal orientations. Currently, cylinders found twice are not merged until we replace the regions with cylinder models in the final polygonalization step. This causes, on average, one cylinder to be oversegmented in every second point cloud. In contrast, not a single cylinder was undersegmented. With respect to missed detections, some of the cylinders in the data set are rather small (radius $< 10\,\mathrm{cm}$) and far away from the sensor (distance $> 2\,\mathrm{m}$). All cylinders missed in the evaluation—on average one in every three point clouds—belong to this class.

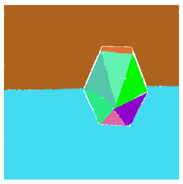
8. Conclusion

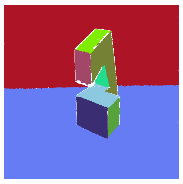
We have presented a fast yet robust approach for segmenting range images and organized point clouds. Using an approximate polygonal mesh reconstruction directly deduced from the image-like structure, we are able to efficiently compute point features such as local surface normals, smooth the measured data using a multilateral filter, and detect planes and other geometric primitives.

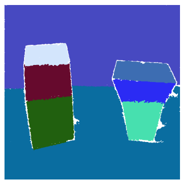
Experimental evaluation has shown that our approach achieves state-of-the-art range image segmentation performance. To achieve real-time processing, we proposed several simplifications and approximations that make the overall segmentation (and primitive detection) algorithm run within milliseconds on a CPU for point clouds acquired by typical RGB-D cameras.

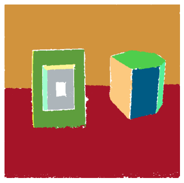

(a) 6/6

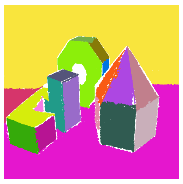

(b) 5/5

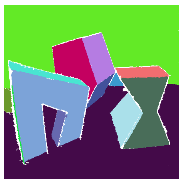

(c) 9/11, 2 misses

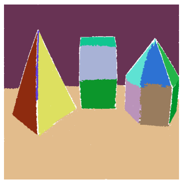

(d) 9/10, 1 miss

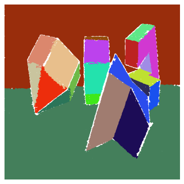

(e) 8/8


(f) 10/14, 4 misses


(g) 18/24, 6 misses


(h) 10/13, 2 over, 1 miss

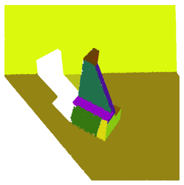

(i) 12/13, 1 over

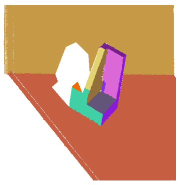

(j) 22/22

	Approach	correctly detected	orientation deviation	over-segmented	under-segmented	missed (not detected)	noise (nonexistent)
Lit. references	USF [8]	8.9 (60.9%)	2.7°	0.4	0.0	5.3	3.6
	WSU [8]	5.9 (40.4%)	3.3°	0.5	0.6	6.7	4.8
	UB [8]	9.6 (65.7%)	3.1°	0.6	0.1	4.2	2.8
	UE [8]	10.0 (68.4%)	2.6°	0.2	0.3	3.8	2.1
	UFPR [8]	**11.0 (75.3%)**	2.5°	0.3	0.1	3.0	2.5
	Oehler et al. [25]	7.4 (50.1%)	5.2°	0.3	0.4	6.2	3.9
Implement.	Georgiev et al. [7]	6.5 (44.2 %)	3.4°	2.3	0.03	5.8	5.0
	Trevor et al. [35]	8.3 (57.0 %)	2.9°	0.6	0.13	5.2	2.1
	Previous work [13]	7.9 (54.1 %)	2.3°	1.4	0.8	5.9	3.5
Ours	Approx. plane seg.	**11.0 (75.3%)**	2.6°	0.4	0.2	2.7	0.3
	Prob. plane seg.	**11.0 (75.3%)**	2.6°	0.4	0.2	2.7	0.3

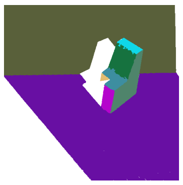
(k) Overall segmentation results on all 30 test images assuming 80 % pixel overlap as in [8].

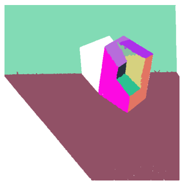
Figure 8: Plane segmentation examples for the SegComp *PERCEPTRON* data set (a-j, segments randomly colored) and detailed results (k) as given in literature (top), reproduced using available implementations (middle) and our approach (bottom). Our approach achieves state-of-the-art performance by correctly segmenting 75.3% of the 14.6 planes. Not correctly found are very small plane segments, e.g., the inner parts of the objects in (f) and (g). In addition, some planes are oversegmented due to noise, e.g., the support plane in (h). The estimated plane normals deviate from ground truth by roughly $(2.5 \pm 1.6)°$.


(a) 10/11, 2 over


(b) 9/10, 2 over

(c) 9/9

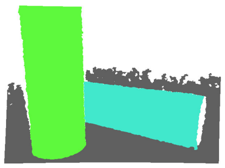

(d) 8/8

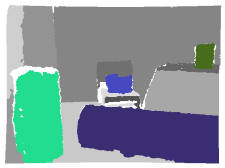

(e) 8/10, 1 over, 1 miss

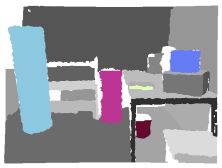
	Approach	correctly detected	orien-tation devia-tion	over-seg-mented	un-der-seg-mented	missed (not de-tected)	noise (nonex-is-tent)
Literature references	USF[8]	12.7 (83.5%)	1.6°	0.2	0.1	2.1	1.2
	WSU [8]	9.7 (63.8%)	1.6°	0.5	0.2	4.5	2.2
	UB [8]	12.8 (84.2%)	1.3°	0.5	0.1	1.7	2.1
	UE [8]	**13.4 (88.1%)**	1.6°	0.4	0.2	1.1	0.8
	OU [8]	9.8 (64.4%)	–	0.2	0.4	4.4	3.2
	PPU [8]	6.8 (44.7%)	–	0.1	2.1	3.4	2.0
	UA [8]	4.9 (32.2%)	–	0.3	2.2	3.6	3.2
	UFPR [8]	13.0 (85.5%)	1.5°	0.5	0.1	1.6	1.4
	Oehler et al. [25]	11.1 (73.0%)	1.4°	0.2	0.7	2.2	0.8
Implement.	Georgiev et al. [7]	6.9 (45.4 %)	2.3°	0.6	1.9	3.6	2.1
	Trevor et al. [35]	9.7 (63.8%)	1.9°	0.8	0.4	3.9	2.8
	Previous work [13]	8.4 (55.1 %)	1.8°	1.2	0.5	4.2	2.3
Ours	Approx. plane seg.	12.2 (80.1%)	1.7°	1.8	0.1	0.9	1.3
	Prob. plane seg.	12.8 (84.2%)	1.4°	0.5	0.1	1.7	2.1

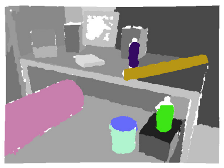
(f) Overall segmentation results on all 30 test images assuming 80 % pixel overlap as in [8].

Figure 9: Plane segmentation examples for the SegComp *ABW* data set (a-e, segments randomly colored) and detailed results (f) as given in literature (top), reproduced using available implementations (middle) and our approach (bottom). Our approach (using approximate plane segmentation tends to over-segment the planes in this data set. On average, 12.2 planes out of 15.2 planes are correctly segmented, while roughly 2 per image are oversegmented. This is primarily caused by the types of occlusions in the *ABW* data set [16].


(a) 2/2


(b) 3/3, 1 non-existent


(c) 4/4


(d) 4/6, 2 over, 1 missed

Approach	correctly detected	over-seg-mented	under-seg-mented	missed (not de-tected)	noise (nonex-istent)
Oehler et al. [25]	1.13 (34.2%)	0.007	0	2.100	6.300
Ours	**2.33 (71.4%)**	0.667	0	0.366	0.100

(e) Overall cylinder detection results on all 30 point clouds (80 % pixel overlap).

Figure 10: Cylinder detection and segmentation in the Kinect cylinder data set by Oehler et al. [25]. Shown are cylinder detections (random colors) and plane detections (random gray tones). A systematic effect of our segmentation approach can be seen in the oversegmentation in (d) where outer (front) and inner (back) part of the cylinder are separate regions.

It remains a matter of future work to exploit the extracted segmented planar patches (and geometric primitives) in further processing steps for purposes such as registration. Further speeding up the approach by parallelizing individual components is also expected to considerably increase efficiency. The implementations of all of the components presented in this paper are (or are going to be) publicly available within the open source Point Cloud Library PCL[3].

Acknowledgments

The authors gratefully acknowledge the financial support from the EU FP7 project ECHORD under Grant No. 231143 and the EU FP7 project STAMINA under Grant No. 610917. We would like to thank Kristiyan Georgiev, Ross T. Creed, and Rolf Lakaemper as well as Alex J. B. Trevor, Suat Gedikli, Radu B. Rusu, and Henrik I. Christensen for providing the implementations of their plane segmentation algorithms. Furthermore, our thanks go to Bastian Oehler and Jörg Stückler for providing the Microsoft Kinect data sets as well as to Adam Hoover, Gillian Jean-Baptiste, Xiaoyi Jiang, Patrick J. Flynn, Horst Bunke, Dimitry B. Goldgof, Kevin Bowyer, David W. Eggert, Andrew Fitzgibbon, and Robert B. Fisher for providing the SegComp data sets and the corresponding

[3]The latest stable release of PCL is available at `http://pointclouds.org`.

evaluation framework.

[1] An, S.-Y., Lee, L.-K., Oh, S.-Y., 2012. Fast incremental 3D plane extraction from a collection of 2D line segments for 3D mapping. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Vilamoura, Portugal, pp. 4530–4537.

[2] Anderson, D., Herman, H., Kelly, A., 2005. Experimental characterization of commercial flash ladar devices. In: Proceedings of the International Conference of Sensing and Technology. Palmerston North, New Zealand.

[3] Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A., 2011. The 3D Hough transform for plane detection in point clouds: A review and a new accumulator design. 3D Research 2, 1–13.

[4] Butt, I. T., Rajpoot, N. M., 2009. Multilateral filtering: a novel framework for generic similarity-based image denoising. In: Proceedings of the IEEE International Conference on Image Processing (ICIP). Cairo, Egypt, pp. 2945–2948.

[5] Cupec, R., Nyarko, E. K., Filko, D., 2011. Fast 2.5D mesh segmentation to approximately convex surfaces. In: Proceedings of the European Conference on Mobile Robots (ECMR). Örebro, Sweden, pp. 49–54.

[6] Fischler, M. A., Bolles, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24 (6), 381–395.

[7] Georgiev, K., Creed, R. T., Lakaemper, R., 2011. Fast plane extraction in 3D range data based on line segments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). San Francisco, CA, USA, pp. 3808–3815.

[8] Gotardo, P. F. U., Bellon, O. R. P., Silva, L., 2003. Range image segmentation by surface extraction using an improved robust estimator. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Madison, WI, USA, pp. 33–38.

[9] Gutmann, J.-S., Fukuchi, M., Fujita, M., 2008. 3D perception and environment map generation for humanoid robot navigation. The International Journal of Robotics Research 27 (10), 1117–1134.

[10] Hähnel, D., Burgard, W., Thrun, S., 2003. Learning compact 3D models of indoor and outdoor environments with a mobile robot. Robotics and Autonomous Systems 44 (1), 15–27.

[11] Harati, A., Gächter, S., Siegwart, R., 2006. Fast range image segmentation for indoor 3D-SLAM. In: Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles (IAV). Toulouse, France.

[12] Holz, D., Behnke, S., 2012. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In: Proceedings of the International Conference on Intelligent Autonomous Systems (IAS). Jeju Island, Korea.

[13] Holz, D., Holzer, S., Rusu, R. B., Behnke, S., 2011. Real-time plane segmentation using RGB-D cameras. In: Proceedings of the RoboCup International Symposium. Istanbul, Turkey.

[14] Holz, D., Schnabel, R., Droeschel, D., Stückler, J., Behnke, S., 2010. Towards semantic scene analysis with time-of-flight cameras. In: Proceedings of the RoboCup International Symposium. Singapore.

[15] Holzer, S., Rusu, R. B., Dixon, M., Gedikli, S., Navab, N., 2012. Real-time surface normal estimation from organized point cloud data using integral images. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Vilamoura, Portugal, pp. 2684–2689.

[16] 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., 1996. An experimental comparison of range image segmentation algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 18, 673–689.

[17] Jiang, X., Bunke, H., 1994. Fast segmentation of range images into planar regions by scan line grouping. Machine Vision and Applications 7, 115–122.

[18] Lee, K.-M., Meer, P., Park, R.-H., 1998. Robust adaptive segmentation of range images. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 200–205.

[19] Magid, E., Soldea, O., Rivlin, E., 2007. A comparison of Gaussian and mean curvature estimation methods on triangular meshes of range image data. Computer Vision and Image Understanding 107 (3), 139–159.

[20] Marton, Z. C., Rusu, R. B., Beetz, M., 2009. On fast surface reconstruction methods for large and noisy datasets. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Kobe, Japan, pp. 3218–3223.

[21] May, S., Droeschel, D., Holz, D., Fuchs, S., Malis, E., Nüchter, A., Hertzberg, J., 2009. Three-dimensional mapping with time-of-flight cameras. Journal of Field Robotics 26 (11-12), 934–965.

[22] Muja, M., Lowe, D. G., 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In: Proceedings of the International Conference on Computer Vision Theory and Application (VISSAPP). Lisbon, Portugal, pp. 331–340.

[23] Nguyen, C. V., Izadi, S., Lovell, D., 2012. Modeling Kinect sensor noise for improved 3D reconstruction and tracking. In: Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT). Zürich, Switzerland, pp. 524–530.

[24] Nüchter, A., Surmann, H., Hertzberg, J., 2003. Automatic model refinement for 3D reconstruction with mobile robots. In: Proceedings of the International Conference on 3-D Digital Imaging and Modeling (3DIM). Banff, Canada, pp. 394–401.

[25] Oehler, B., Stückler, J., Welle, J., Schulz, D., Behnke, S., 2011. Efficient multi-resolution plane segmentation of 3D point clouds[4]. In: Proceedings of the International Conference on Intelligent Robotics and Applications (ICIRA). Aachen, Germany, pp. 145–156.

[26] Pathak, K., Vaskevicius, N., Birk, A., 2009. Revisiting uncertainty analysis for optimum planes extracted from 3D range sensor point-clouds. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Kobe, Japan, pp. 1631–1636.

[27] Poppinga, J., Vaskevicius, N., Birk, A., Pathak, K., 2008. Fast plane detection and polygonalization in noisy 3D range images. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Nice, France, pp. 3378–3383.

[28] Rabbani, T., van den Heuvel, F. A., Vosselman, G., 2006. Segmentation of point clouds using smoothness constraint. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 36, 248–253.

[29] Rusu, R. B., Blodow, N., Marton, Z. C., Beetz, M., 2009. Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in human environments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). St. Louis, MO, USA, pp. 1–6.

[30] Schnabel, R., Wahl, R., Klein, R., 2007. Efficient RANSAC for point-cloud shape detection. Computer Graphics Forum 26 (2), 214–226.

[31] Silva, L., Bellon, O., Gotardo, P., 2002. A global-to-local approach for robust range image segmentation. In: Proceedings of the IEEE International Conference on Image Processing (ICIP). Rochester, NY, USA, pp. 773–776.

[4]Note: not all results referenced here have been published in the paper but all results can be found in Oehler's diploma thesis at `http://www.ais.uni-bonn.de/theses/Bastian_Oehler_Diplomarbeit_08_2011.pdf`

[32] Smisek, J., Jancosek, M., Pajdla, T., 2011. 3D with Kinect. In: Workshop Proceedings of the IEEE International Conference on Computer Vision (ICCV Workshops). Barcelona, Spain, pp. 1154–1160.

[33] Stückler, J., Steffens, R., Holz, D., Behnke, S., 2011. Real-time 3D perception and efficient grasp planning for everyday manipulation tasks. In: Proceedings of the European Conference on Mobile Robots (ECMR). Örebro, Sweden, pp. 177–182.

[34] Torr, P., Zisserman, A., 2000. MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding 78, 138–156.

[35] Trevor, A. J. B., Gedikli, S., Rusu, R. B., Christensen, H. I., 2013. Efficient organized point cloud segmentation with connected components. In: Proceedings of the 3rd Workshop on Semantic Perception, Mapping and Exploration (SPME) at ICRA. Karlsruhe, Germany.

[36] Vosselman, G., Gorte, B. G. H., Sithole, G., Rabbani, T., 2004. Recognising structure in laser scanner point clouds. Information Sciences 46 (8), 1–6.