# A Maximum Likelihood Approach to Extract Finite Planes from 3-D Laser Scans

Alexander Schaefer, Johan Vertens, Daniel Büscher, Wolfram Burgard

***Abstract*— Whether it is object detection, model reconstruction, laser odometry, or point cloud registration: Plane extraction is a vital component of many robotic systems. In this paper, we propose a strictly probabilistic method to detect finite planes in organized 3-D laser range scans. An agglomerative hierarchical clustering technique, our algorithm builds planes from bottom up, always extending a plane by the point that decreases the measurement likelihood of the scan the least. In contrast to most related methods, which rely on heuristics like orthogonal point-to-plane distance, we leverage the ray path information to compute the measurement likelihood. We evaluate our approach not only on the popular SegComp benchmark, but also provide a challenging synthetic dataset that overcomes SegComp's deficiencies. Both our implementation and the suggested dataset are available at [1].**

## I. Introduction

Most man-made environments like factory floors, offices, and households are primarily polyhedral: Their surfaces can be described by a set of finite planes. Robots navigating these types of environments often rely on 3-D laser range finders, which capture up to millions of reflections per second. Plane extraction methods take these highly redundant raw sensor data and reduce them to the parameters of the underlying planes.

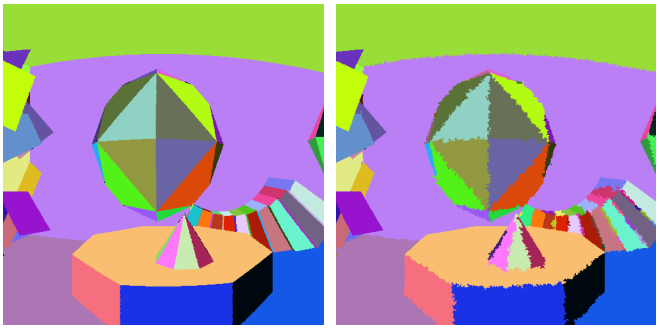
Plane extraction has obvious benefits. For example, processing a few planes instead of millions of points requires considerably less computational effort and smaller memory footprint. Plane extraction may also increase accuracy in tasks like scan matching and sensor calibration, and it enables applications like model reconstruction and object detection in the first place.

The presented method, dubbed probabilistic plane extraction (PPE), extends our recent work on polyline extraction from 2-D laser range scans [2] to three dimensions. Essentially, PPE is a maximum likelihood approach based on agglomerative hierarchical clustering. In the beginning, PPE represents the scan by a large set of planes – one for every reflection – and then iteratively merges them, in each step choosing the subset whose merger maximizes the measurement likelihood of the whole scan, until a specified stopping criterion is met, for example a certain number of planes. Figure 1b shows an exemplary segmentation result.

Our approach distinguishes itself from the large body of related work in two respects. First, all methods surveyed in the following resort to heuristics like orthogonal distance between ray endpoint and plane when estimating the measurement likelihood of a scan conditioned on a set of planes. Instead, PPE accounts for the true ray path from start to end. This more accurate sensor model leads to more accurate results, as demonstrated by our experiments. Second, due to its probabilistic formulation, PPE requires only one robust parameter to control the granularity of the extracted planes. In contrast, some of the surveyed methods need up to a dozen carefully tuned parameters in order to obtain reasonable results.

This work has been partially supported by Samsung Electronics Co. Ltd. under the GRO program.

All authors are with the Department of Computer Science, University of Freiburg, Germany.
{aschaef, vertensj, buescher, burgard}@cs.uni-freiburg.de



(a) Ground-truth segmentation. (b) PPE segmentation.

Fig. 1: Ground-truth segmentation of an organized $500 \times 500$ point cloud taken from the suggested SynPEB dataset and segmentation result of PPE, the proposed method.

## II. Related Work

This section provides an overview over the state of the art considering plane extraction from 3-D lidar scans. In our methodology, we distinguish four classes of approaches: approaches based on region growing, on clustering, on random sample consensus (RANSAC), and approaches based on the Hough transform.

In a nutshell, region growing first selects some so-called seed points from the input point cloud, which are then grown into regions by iteratively adding all neighboring points that pass a set of criteria. Hoover et al. [3], for instance, select the points with the highest local planarity score as seeds. During the growing process, they add all adjacent points to the regions that do not exceed a specified maximum difference of normals, Euclidean distance, and orthogonal distance. In contrast, Hähnel et al. [4] choose seeds at random and grow planar polygons by including all neighboring points that, among other criteria, do not push the mean squared error of the resulting plane over a given limit.

Deschaud et al. [5] propose an adaption of region growing to large noisy datasets. They compensate for noise by introducing a filter that improves the estimation of endpoint normals, select seeds based on local planarity, and employ a voxel-based variant of region growing. Nurunnabi et al. [6], in turn, address noise by computing endpoint features like normals and curvature via a robust variant of principal component analysis (PCA). In another take on plane extraction from noisy point clouds, Dong et al. [7] combine region growing with energy optimization. They first oversegment the point cloud into multiscale supervoxels, compute associated features, and then apply region growing. Finally, they refine the resulting planes by means of global energy optimization, where the global energy is defined as the sum of geometric errors, spatial coherence, and the total number of planes.

Holz et al. [8] focus on fast plane extraction for time-sensitive applications like object tracking. Their method computes normal and curvature estimates not directly based on the point cloud, but based on an approximate mesh. CAPE, an algorithm developed by Proença et al. [9], achieves even higher plane extraction rates at the expense of reduced accuracy. First, the algorithm creates a low-resolution grid, pools the points in each cell, and applies PCA to each cell. CAPE then subjects the cells along with the features obtained by PCA to region growing, fits planes to the resulting regions, and eventually refines the segmentation.

Inspired by the observation that every sequence of points in a laser scan that describes a line is caused by a planar surface, Jiang et al. [10], Hoover et al. [3], and Cabo et al. [11] apply region growing to line segments instead of points.

As opposed to region growing, clustering extracts planes without the need to find suitable seed points. In an early work on clustering-based plane extraction, for example, Hoffman et al. [12] segment a range scan into planar, concave, and convex surface patches, and then combine compatible patches to detect contiguous faces. Instead of segmenting surface patches first, Trevor et al. [13] operate directly on the endpoints of an organized scan obtained from a projective range sensor. They assign the same label to adjacent points if the difference of their normals and their orthogonal distance falls below a given threshold, and subsequently extract planes by clustering points with the same labels.

Feng et al. [14] present a clustering algorithm that aims at extracting planes from an organized point cloud with minimal latency. It divides the point cloud uniformly into rectangular point groups, discards all non-planar groups, and subjects the remaining groups to agglomerative hierarchical clustering, using the mean squared orthogonal point-to-plane fitting error as clustering metric. Eventually, it refines the extracted coarse planes by region growing. Marriott et al. [15] also cluster groups of coplanar points based on mean squared error, but instead of using a regular grid to define initial point groups, they propose an expectation-minimization algorithm that fits a Gaussian mixture model to the points.

Pham et al. [16] combine clustering and region growing. They use region growing to oversegment the point cloud and then merge the resulting plane hypotheses via clustering, in each step minimizing an energy function that favors mutually parallel or orthogonal plane pairs.

RANSAC, initially developed by Fischler et al. [17], is a versatile iterative model fitting algorithm. When applied to plane extraction, it essentially selects three laser endpoints at random, fits a plane to them, searches for all points within a certain orthogonal distance, and determines the plane's fitness based on the corresponding point-to-plane distances. This process is repeated until the algorithm finds a plane that satisfies a given minimal fitness.

Several works improve on standard RANSAC in order to overcome its deficiencies. Gotardo et al. [18], for example, derive a robust estimator that counteracts RANSAC's tendency to disregard small regions. Gallo et al. [19] address the problem of RANSAC often connecting nearby patches that are actually unconnected, for example at steps, curbs, or ramps. They call their solution CC-RANSAC, a RANSAC variant that considers only the largest connected component of inliers to evaluate the fitness of a candidate plane. By combining RANSAC with conformal geometric algebra, Sveier et al. [20] perform the least squares fitting necessary to assess the fitness of a plane hypothesis analytically instead of numerically. Alehdaghi et al. [21] present a highly parallelized GPU implementation of RANSAC for plane extraction that yields a significant speedup compared to executing the algorithm on the CPU.

Another general model fitting method, the Hough transform computes for each point in the discretized space of model parameters the fitness of the associated model instance given the data. Vosselman et al. [22] describe how to apply this method to the problem of plane extraction from 3-D point clouds. They also propose a variant of the Hough transform that leverages point normals to increase efficiency. Oehler et al. [23] present a multi-resolution approach based on both the Hough transform and RANSAC. They extract surface normals from the point cloud at different resolutions and use the Hough transform to split non-planar patches into groups of coplanar points. After clustering adjacent coplanar patches, they employ RANSAC to fit planes to the resulting connected components. For a review of further flavors of Hough transform-based plane extraction, the reader is referred to the review composed by Borrmann et al. [24].

## III. Approach

In this work, we present an approach to extract finite planes from organized 3-D lidar scans, called probabilistic plane extraction (PPE). PPE is a maximum likelihood estimation technique based on agglomerative hierarchical clustering. As a maximum likelihood estimation technique, it searches for the set of planes that maximizes the measurement probability of the given laser scan. As an agglomerative clustering method, it attempts to find this set by creating a plane for each reflection first. This plane explains the corresponding reflection perfectly. PPE then reduces the number of planes by iteratively merging the set of adjacent planes whose merger maintains the highest measurement

likelihood of the scan. Clustering ends as soon as a given stopping criterion is met.

In the following, we first introduce the probabilistic sensor model, on the basis of which we then formulate plane extraction as a maximum likelihood estimation problem. We describe in detail how our agglomerative hierarchical clustering algorithm solves this optimization problem, and finally give a walkthrough of the pseudocode.

### A. Probabilistic Sensor Model

The sensor model tells the measurement probability of a 3-D lidar scan given a set of planes. We denote the scan $Z \coloneqq \{z_k\}$, where $k \in \{1, 2, \dots, K\}$ represents the index of a laser ray. A single laser measurement $z \coloneqq \{s, v, r\}$ is composed of two three-element Cartesian vectors and a scalar: the starting point $s$ of the ray, the normalized direction vector $v$, and the ray length $r$. The set of finite planes $L \coloneqq \{l_j\}$ extracted from the scan consists of a total of $J$ elements. Each plane is represented by a three-element Cartesian support vector $x$, a three-element Cartesian normal vector $n$, and a set $Q$ of ray indices: $l \coloneqq \{x, n, Q\}$. While $x$ and $n$ define the location and orientation of the plane, $Q$ determines its extent. This representation can not only handle convex planes, but also concave planes or planes with holes.

With these definitions, we can now define the measurement model. Most lidar sensors exhibit approximately normally distributed noise in radial direction and relatively small angular noise. Consequently, we neglect angular noise and model the distribution of the measured length of a single ray conditioned on a set of planes as a Gaussian probability density function centered at the true ray length:

$$p(z \mid L) = \mathcal{N}(r; \hat{r}(s, v, L), \sigma^2). \tag{1}$$

Here, the function $\hat{r}(s, v, L) \in \mathbb{R}^+$ computes the distance between the starting point of the ray and the first intersection of its axis and all planes in $L$. The standard deviation $\sigma$ of the radial noise is usually a function of multiple parameters such as sensor device, reflecting surface, and temperature.

By assuming independence between the individual laser rays, we can derive the measurement probability of the whole scan from equation (1) as

$$p(Z \mid L) = \prod_{k=1}^{K} p(z_k \mid L).$$

To our knowledge, we are the first to apply the above sensor model to the problem of plane extraction. In contrast, most surveyed works model the measurement probability of a single ray as a zero-centered normal distribution over the shortest distance between the measured ray endpoint and the nearest plane. This heuristic does not account for the ray path, which leads to two undesired effects. First, the nearest plane is not always the one that intersects the ray. Second, the accuracy of the computed distance strongly depends on the incidence angle of the ray.

### B. Maximum Likelihood Estimation

Given the above sensor model, we formulate plane extraction as the following maximum likelihood estimation problem: Find the set of planes $L^*$ that maximizes the measurement probability of the whole scan $p(Z \mid L)$. The solution is trivial: For each reflection in the laser scan, create a tiny plane that is not parallel to the ray and that intersects the ray at the measured ray length $r$. This solution, however, is merely a different representation of the raw lidar data. In order to extract meaningful planes from the scan, we need to reduce the number of planes by constraining the optimization problem. The constraint quantifies the application-dependent compromise between the memory requirements and the accuracy of the produced planes. Embedded applications, for example, might focus on extracting only few planes to maintain minimal memory footprint, while offline mapping systems might favor high accuracy at the expense of large numbers of planes. For the following derivation, we choose the maximum number of planes $J_{\max}$ as constraint parameter. Note, however, that our approach allows us just as well to use arbitrary parameters like the maximum mean squared error of the ray radii or the Akaike Information Criterion [25].

The resulting problem statement reads as follows: Given the maximum number of planes $J_{\max}$, find the set of planes $L^*$ that, among all other plane maps with at most $J_{\max}$ elements, yields the highest measurement likelihood. Formally, we are confronted with the constrained least squares optimization problem

$$\begin{aligned} L^* &= \operatorname*{argmax}_L p(Z \mid L) \Big|_{J(L) \le J_{\max}} \\ &= \operatorname*{argmin}_L -\log\Big(p(Z \mid L)\Big) \Big|_{J(L) \le J_{\max}} \\ &= \operatorname*{argmin}_L \underbrace{\sum_{k=1}^{K} \Big(r_k - \hat{r}(s_k, v_k, L)\Big)^2}_{=: E(Z, L)} \Big|_{J(L) \le J_{\max}} \\ &= \operatorname*{argmin}_L E(Z, L) \Big|_{J(L) \le J_{\max}} \end{aligned} \tag{2}$$

Here, $J(L)$ is a function that determines the number of planes in $L$. The transition from the second to the third line in equation (2) implies our assumption that all rays exhibit the same radial noise. Hereafter, we will refer to $E$ simply as the error of the set of planes $L$.

Solving (2) is primarily a combinatorial problem. Even if we knew the parameters $\{x_j\}$ and $\{n_j\}$ of the planes, we would still not know the data associations $\{Q_j\}$, i.e. which rays belong to which plane. Exhaustively searching the space of all data associations for the combination that maximizes the measurement probability quickly leads to combinatorial explosion even for small $J_{\max}$. PPE solves this problem via agglomerative hierarchical clustering, as delineated in the next section.

### C. Agglomerative Hierarchical Clustering

In its generic form, agglomerative hierarchical clustering builds clusters from bottom up: The algorithm first assigns each observation its own cluster and then iteratively merges adjacent pairs of clusters. In each iteration, it decides which pair to merge based on a greedy strategy, always optimizing a specific metric.

Transferred to our case, observations correspond to reflected laser rays, clusters correspond to planes, and the metric the algorithm strives to maximize is the measurement probability $p(Z \mid L)$, which is equivalent to minimizing the error $E(Z, L)$. Consequently, in the first step, which assigns each observation its own cluster, we assign each laser reflection its own plane. As mentioned above, this plane is not parallel to the ray and intersects the ray at its measured length $r$. In the following, we call such a plane atomic. Quite arbitrarily, we define the support vector of an atomic plane as the endpoint $s + rv$ of the corresponding ray, and the normal vector as the ray direction vector $v$. As opposed to atomic planes, regular planes represent not one, but three or more rays. Therefore, their parameters need to be fitted to the data.

Starting from this trivial maximum likelihood solution, PPE iteratively reduces the number of planes to $J_{\max}$ by merging adjacent planes, regardless of whether they are atomic or regular. With each merger, the measurement likelihood of the whole scan $p(Z \mid L)$ decreases, whereas the error $E(Z, L)$ increases by

$$e \coloneqq E(Z, L'') - E(Z, L') \geq 0, \tag{3}$$

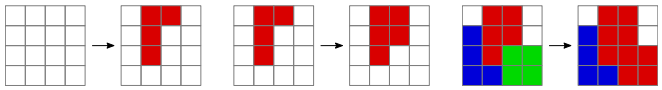
where $L'$ and $L''$ denote the set of planes before and after the merger. Greedy as it is, PPE always opts for the merger that incurs the least error increment, which is equivalent to choosing the merger that maintains maximum measurement likelihood.

In the formulation above, clustering will not yield the desired result yet. This is due to ambiguities in the decision process: The error increment corresponding to merging two atomic planes is always zero, because every pair of reflections can be perfectly explained by a single plane. Therefore, given multiple atomic planes, PPE cannot decide which pair to merge. The same applies to merging three atomic planes. Creating a regular plane out of four atomic planes, however, leads to an overdetermined system of equations, hence the regular plane must be fitted to the four reflections, and the corresponding fitting error constitutes the error increment

$$e_{\text{crt}}(Z, Q) \coloneqq \min_{x,n} E(\{z_q\}, \{x, n, Q\}), \tag{4}$$

where $Q$ denotes the set of ray indices, and where $q \in Q$.

In order to find the combination of four atomic planes that yields the minimum error increment, PPE needs to assess the fitting errors corresponding to all possible combinations. For an atomic plane that resides somewhere in the middle of the grid of laser rays, there are 17 valid ways to combine it with three of its 4-connected neighbors, forming so-called tetrominoes: one O-shaped tetromino, four T-tetrominoes, four Z-tetrominoes, and eight L-tetrominoes. I-shaped tetrominoes are invalid, because fitting a plane to four endpoints in a straight line again yields ambiguous results.



(a) Creating a regular plane from four atomic planes.

(b) Extending a regular plane by an atomic plane.

(c) Merging two regular planes.

Fig. 2: Exemplary instances of all three classes of actions PPE can take during clustering in order to reduce the number of planes. White fields stand for atomic planes, fields of the same color except white denote regular planes.

Once the first regular planes emerge, we can identify two more classes of clustering actions apart from merging tetrominoes: extending a regular plane by an atomic plane, and merging two regular planes. Figure 2 illustrates all three classes. In each clustering step, PPE must determine the error increment of every possible action, find the one that incurs the least error increment, and merge the respective planes. In the following, we will derive the error increments for each action class.

The error increment $e_{\text{crt}}$ of merging four atomic planes has already been defined in equation (4). The error increment of extending a regular plane by an atomic plane amounts to the difference

$$e_{\text{ext}}(Z, Q, k) \coloneqq e_{\text{crt}}(Z, Q \cup k) - e_{\text{crt}}(Z, Q),$$

where $Q$ denotes the indices of the rays of the regular plane, and where $k$ is the index of the ray corresponding to the atomic plane. Merging two regular planes indexed $i$ and $j$ adds

$$e_{\text{mrg}} \coloneqq e_{\text{crt}}(Z, Q_i \cup Q_j) - e_{\text{crt}}(Z, Q_i) - e_{\text{crt}}(Z, Q_j)$$

to the total error $E(Z, L)$.

With all the individual terms of PPE well defined, we now summarize the algorithm by providing the pseudocode.

### D. Probabilistic Plane Extraction

Algorithm 1 shows the pseudocode of a straightforward PPE implementation. Line 1 initializes the set of atomic planes. The function $\text{crt}(Z, L)$ in line 2 loops over all valid tetrominoes of atomic planes and returns the minimum error $e_{\text{crt}}$ along with the associated indices $Q_{\text{crt}}$. As there are no regular planes which could be extended or merged at this point, line 3 sets the corresponding error increments $e_{\text{ext}}$ and $e_{\text{mrg}}$ to infinity. After these initializations, the algorithm starts iteratively reducing the number of planes. In the first iteration, it always creates a regular plane out of four atomic ones. This means first it first adds the new plane to the map (line 6) and then removes the merged atomic planes (line 7). Here, the function $\text{fit}(Z, Q)$ fits a plane $l^*$ to the rays indexed by $Q$:

$$l^* \coloneqq \text{fit}(Z, Q) \coloneqq \Big\{ \operatorname*{argmin}_{x,n} E(\{z_q\}, \{x, n, Q\}), Q \Big\},$$

**Algorithm 1:** Probabilistic Plane Extraction

**Data:** $Z$, $J_{\max}$
**Result:** $L$

```
1  L ← {s_k + r_k v_k, v_k, k}, k ∈ {1, 2, ..., K}
2  (e_crt, Q_crt) ← crt(Z, L)
3  e_ext ← e_mrg ← ∞
4  while J(L) > J_max do
5      if e_crt = min(e_crt, e_ext, e_mrg) then
6          L ← L ∪ fit(Z, Q_crt)
7          L ← rma(L, Q_crt)
8          else if e_ext = min(e_crt, e_ext, e_mrg) then
9              L_j ← fit(Z, Q_j ∪ k)
10             L ← rma(L, {k})
11             else
12                 L_j ← fit(Z, Q_i ∪ Q_j)
13                 L ← L \ L_i
14             end
15         end
16     end
17     (e_crt, Q_crt) ← crt(Z, L)
18     (e_ext, j, k) ← ext(Z, L)
19     (e_mrg, i, j) ← mrg(Z, L)
20 end
```

whereas $\mathrm{rma}(L, Q)$ in line 7 removes the atomic planes indexed by $Q$ from $L$ and returns the updated plane set. After every manipulation of the plane map, lines 17 to 19 recompute the error increments of all merging options. To that end, $\mathrm{ext}(Z, L)$ iterates over all possible extensions of all regular planes in $L$ and finds the minimum error increment $e_{\text{ext}}$ associated with extending plane $j$ by ray $k$. Similarly, $\mathrm{mrg}$ evaluates for all pairs of neighboring regular planes the hypothetical error increments incurred by merging them and returns the minimum $e_{\text{mrg}}$, which corresponds to merging planes $i$ and $j$. Lines 9 and 10 update the map during an extension step, while lines 12 and 13 come into play when two regular planes are merged.

For the sake of clarity, algorithm 1 is not optimized. For example, lines 17 to 19 unnecessarily recompute all error terms in each iteration. For an optimized version of PPE, please refer to our MATLAB implementation, publicly available at [1]. In addition to several algorithmic optimizations, it features optional GPU acceleration, multiple stopping criteria, and a geometric outlier filter.

## IV. Experiments

In order to compare the results provided by the presented plane extraction algorithm to the state of the art, we conduct two series of experiments. In the first series, we evaluate PPE using the popular SegComp plane extraction benchmark [3]. The deficiencies of this dataset motivate us to create SynPEB, the first publicly available synthetic plane extraction benchmarking dataset, on which we base the second experiment series. Below, we describe both benchmarks and the corresponding experiments in detail.

SegComp comprises two collections of organized point clouds, which depict compositions of polyhedral objects on a tabletop. They were recorded by an ABW structured light sensor and by a Perceptron laser scanner, respectively. Due to the fact that our measurement model, defined in equation (1), is specifically designed for laser sensors, we evaluate our method on the Perceptron collection only. This dataset is divided into 10 training scans and 30 testing scans. We use the former to determine the optimum values of $e$ and $d$, the two parameters of the specific PPE version we use in both experiment series. The parameter $e$, defined in equation (3), denotes the maximum admissible error increment in a clustering step and serves as stopping criterion. While section III suggests the minimum number of planes as stopping criterion, we choose $e$ in both experiment series, because the numbers of planes differ across the scans. Moreover, in order to compensate for the high level of noise present in all Perceptron scans, we incorporate a geometric outlier filter in PPE, which prevents clustering neighboring points if their Cartesian distance exceeds a certain threshold $d$. To find suitable values for both parameters, we maximize the fraction of correctly segmented planes over a regular grid in $e$ and $d$. Table I displays the results of this parameter sweep.

The upper part of table II shows the corresponding experimental results for PPE and compares them to all previous works evaluated on the Perceptron dataset using the performance metrics defined by Hoover et al. [3]. In order to increase the relevance of the results, we suggest two additional metrics: the $k$-value and the $\mathrm{RMSE}$. The $k$-value is defined as

$$k \coloneqq \frac{\sum_{j=1}^{J(L)} \hat{K}(l_j)}{K}, \tag{5}$$

where $\hat{K}(l)$ is a function that takes an extracted plane $l$ as input, checks if this plane is correctly segmented using the $80\,\%$ threshold proposed by Hoover et al., and returns the number of points of the corresponding ground truth plane. If the input plane is not correctly segmented, the function returns zero. In this way, $k$ indicates the portion of the point cloud that the algorithm correctly segments into planes. The root mean squared error $\mathrm{RMSE}$, defined as

$$\mathrm{RMSE} \coloneqq \sqrt{\frac{E(Z, L)}{J(L)}}, \tag{6}$$

complements $k$ by providing an estimate of how accurately the extracted planes represent the point cloud.

In addition to quoting the numbers of previous works and stating our results for PPE, we evaluate MSAC and PEAC. MSAC is a simple baseline approach we implemented based on the RANSAC variant proposed by Torr et al. [27]. Beginning with the input point cloud, this method iteratively detects a plane and removes the inlier points from the cloud until a specified fraction of the original number of points remains. PEAC – plane extraction using agglomerative

| Method | Parameter | Unit | SegComp | | | SynPEB | | |
|---|---|---|---|---|---|---|---|---|
| | | | Range | Opt | $f^*[\%]$ | Range | Opt | $f^*[\%]$ |
| PEAC [26] | arccos(similarityTh_merge) | [deg] | $[20, 80]_7$ | 20 | 62.7 | $[3, 40]_7$ | 21.5 | 32.7 |
| | arccos(similarityTh_refine) | [deg] | $[10, 50]_7$ | 23.3 | | $[3, 30]_7$ | 12 | |
| | depthAlpha | [1] | $[0.003, 0.300]_7$ | 0.102 | | $[0.003, 0.010]_7$ | 0.0088 | |
| | depthChangeTol | [1] | $[0, 0.030]_7$ | 0.005 | | $[0.005, 0.040]_7$ | 0.040 | |
| | minSupport | [1] | $[50, 1500]_7$ | 292 | | $[20, 2000]_7$ | 20 | |
| MSAC [27] | $a$ | [mm] | $[0.5, 20]_{40}$ | 5 | 24.5 | $[2, 80]_{40}$ | 34 | 11.7 |
| | $b$ | [%] | $[1, 30]_{30}$ | 4 | | $[1, 30]_{30}$ | 3 | |
| PPE | $d$ | [cm] | $[0.4, 4]_{10}$ | 2 | 83.6 | $[2, 20]_{10}$ | 18 | 78.6 |
| | $\sqrt{e}$ | [cm] | $[2, 100]_{50}$ | 46 | | $[2, 100]_{50}$ | 22 | |

TABLE I: Results of the parameter sweeps performed in both experiment series on the training scans of the corresponding datasets. The notation $[x, y]_n$ stands for the discrete set of parameters $\{x, x + 1\frac{y-x}{n-1}, x + 2\frac{y-x}{n-1}, \ldots, y\}$ for which $f$, the fraction of correctly segmented planes averaged over all training scans, was computed. All parameter bounds were carefully selected via random search. The values $f^*$ and "Opt" tell the maximum $f$ and the associated parameter values, respectively. Although the open-source implementation of PEAC [26] contains 18 parameters, in our experiments, only the five of them listed above had a significant effect on the quality of the segmentation result. For MSAC, $a$ denotes the maximum orthogonal distance between a plane and a plane inlier, while $b$ is the maximum fraction of points that are not assigned to a plane.

clustering – refers to the open-source implementation [26] Feng et al. provide to complement their paper [14]. Using this implementation, we are not able to exactly replicate the SegComp results they quote in their paper. Nevertheless, we state our findings for SegComp in order to establish comparability between our PEAC results across both experiment series. Analogously to PPE, we determine the optimum parameters for MSAC and PEAC via grid search – see table I. Even with these parameters, both MSAC and PEAC return a single false plane detection when processing all testing scans of SegComp, which leads to exploding RMSE-values. To mitigate this effect, the RMSE-values in table II are based on all planes with $\mathrm{RMSE} \leq 10\,\mathrm{m}$ each.

Although PPE is designed for maximum accuracy, our method achieves only average results on SegComp. The reasons for that lie in the peculiarities of the dataset, which we will illustrate by figure 3. To begin with, figure 3a reveals that the rays that hit an object face at an obtuse angle are much more strongly affected by noise than rays with acute incidence angles, creating the impression that faces with obtuse incidence angles extend in a curved fashion beyond their borders. This fact by itself does not pose a problem, because every plane extraction method should be able to tolerate certain amounts of noise. What negatively affects the evaluation, however, is the ground-truth segmentation. Instead of identifying outliers as such, almost all measurements are assigned to object faces, regardless of the distance between the respective points and the actual face. Obviously, the labeling was performed on a 2-D projection of the point cloud like the one shown in figure 3b rather than on a 3-D representation. Another issue with SegComp becomes apparent when closely inspecting the ground plane: Labeling is based on the geometry of the underlying objects, not on the output of the miscalibrated sensor. The khaki tabletop plane and the purple topside of the octagon in the point cloud in figure 3a, for example, exhibit kinks due to systematic errors in the lidar calibration. The labelers, knowing that these planes were flat, labeled both as contiguous planes. PPE, without knowledge about the real scene, splits each plane into two, as shown in figure 3c. Although desirable, this behavior results in the highest oversegmentation rate among all methods compared on SegComp and decreases the percentage of correctly segmented planes as well as the $k$-value. In order to prove that the ground-truth labeling of the Perceptron dataset is indeed not optimal given the original number of labels per scan, we compare the RMSE-values of the ground-truth segmentation to those of PPE. This time, PPE is configured to extract as many planes from a scan as there are present in the ground truth. On average, the resulting RMSE-values are $3.2\,\%$ lower than those corresponding to ground truth. A $t$-test over all scans yields a $p$-value of $12.9\,\%$, which means that the probability of PPE returning a more accurate segmentation than ground truth is as high as $87.1\,\%$. Similarly to the ground-truth segmentation, the ground-truth angles between adjacent planes were presumably determined based on the underlying data, too: They are provided as integers rather than as floating-point numbers.

As the aforementioned problems with SegComp bias the evaluation and because there is no publicly available alternative, we create a synthetic plane extraction benchmark dataset, in short SynPEB, which we use as the basis of the second experiment series. Like our implementation of PPE and the code required to run all our experiments, both the SynPEB scans and the sampling engine can be downloaded at [1]. The SynPEB world consists of a room of approximately

| Method | $f$ [%] | $k$ [%] | RMSE [mm] | $\alpha$ [°] | $n_o$ | $n_u$ | $n_m$ | $n_s$ |
|---|---|---|---|---|---|---|---|---|
| SegComp Perceptron dataset | | | | | | | | |
| USF [3] | 60.9 | – | – | 2.7 | 0.4 | 0.0 | 5.3 | 3.6 |
| WSU [3] | 40.4 | – | – | 3.3 | 0.5 | 0.6 | 6.7 | 4.8 |
| UB [3] | 65.7 | – | – | 3.1 | 0.6 | 0.1 | 4.2 | 2.8 |
| UE [3] | 68.4 | – | – | 2.6 | 0.2 | 0.3 | 3.8 | 2.1 |
| UFPR [18] | 75.3 | – | – | 2.5 | 0.3 | 0.1 | 3.0 | 2.5 |
| Oehler et al. [23] | 50.1 | – | – | 5.2 | 0.3 | 0.4 | 6.2 | 3.9 |
| Holz et. al. [8] | 75.3 | – | – | 2.6 | 0.4 | 0.2 | 2.7 | 0.3 |
| RPL-GMR [15] | 72.4 | – | – | 2.5 | 0.3 | 0.3 | 3.0 | 2.0 |
| Feng et al. [14] | 60.9 | – | – | 2.4 | 0.2 | 0.2 | 5.1 | 2.1 |
| PEAC [26] | 48.6 | 91.3 | 2.6 | 2.6 | 0.0 | 0.1 | 7.1 | 2.0 |
| MSAC [27] | 18.5 | 76.7 | 3.4 | 3.9 | 0.1 | 0.2 | 11.3 | 3.4 |
| PPE (proposed) | 60.7 | 61.2 | 2.9 | 2.8 | 1.4 | 1.1 | 1.5 | 2.3 |
| SynPEB dataset | | | | | | | | |
| PEAC [26] | 29.1 | 60.4 | 28.6 | – | 0.7 | 1.0 | 26.7 | 7.4 |
| MSAC [27] | 7.3 | 35.6 | 34.3 | – | 0.3 | 1.0 | 36.3 | 10.9 |
| PPE (proposed) | 73.6 | 77.9 | 14.5 | – | 1.5 | 1.1 | 7.1 | 16.5 |

TABLE II: Results of both experiment series. The header variables $f$ and $\alpha$ denote the fraction of correctly segmented planes and the mean angular deviation, averaged over all testing scans, while $n_o$, $n_u$, $n_m$, and $n_s$ represent the absolute numbers of oversegmented, undersegmented, missing, and spurious planes compared to the ground-truth segmentation. The metrics $k$ and $\mathrm{RMSE}$ are defined in equation (5) and (6), respectively. On average, each scan of the SegComp dataset contains $14.6$ ground-truth planes, while each scan of the SynPEB dataset is composed of $42.6$ planes.

$6\,\mathrm{m} \times 7\,\mathrm{m} \times 3\,\mathrm{m}$ populated with various polyhedral objects, resulting in $42.6$ planes of different shapes and sizes per scan. Analogously to SegComp, we divide the dataset into $10$ training scans and $30$ testing scans, provided as organized point clouds of $500 \times 500$ measurements. These scans are affected by normally distributed angular noise with standard deviation $\sigma_{\text{ang}} = 1\,\mathrm{mdeg}$ and by normally distributed radial noise with $\sigma_{\text{rad}} = 20\,\mathrm{mm}$. Figure 1 conveys an intuition of what a SynPEB scan looks like.

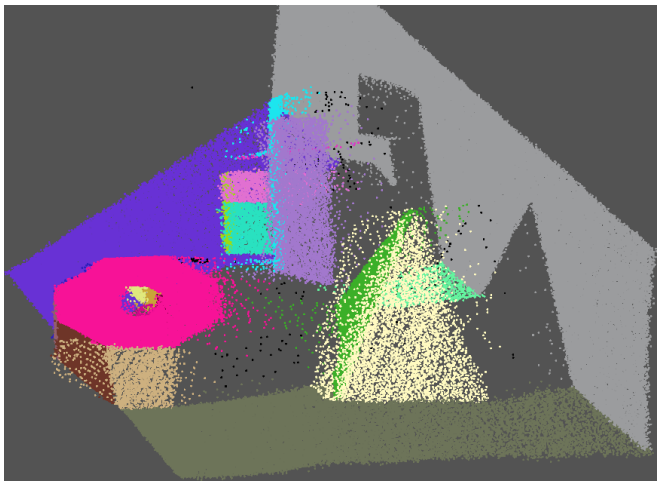
The lower part of table II shows the plane extraction results for PEAC, MSAC, and PPE on SynPEB. For the other approaches, there is no working implementation publicly available. When comparing the results across datasets, we observe that the fraction of planes detected by both PEAC and MSAC is considerably lower on SynPEB than on SegComp. As indicated by the high numbers of missed planes, the most likely cause is the challenging nature of SynPEB: At almost identical resolutions, SynPEB contains almost three times as many planes per scan as SegComp. Additionally, the variations in plane sizes are larger than in SegComp. Nevertheless, both the percentage of correctly identified planes and the $k$-value of the results produced by PPE have increased significantly. For an illustration of a corresponding segmentation output, see figure 1b. The $\mathrm{RMSE}$-value for PPE is $28\,\%$ lower than the radial noise, demonstrating that the method is able leverage the high number of data points per plane to accurately reconstruct the underlying data. The respective value on SegComp is comparatively high, because the parameter optimization was compromised by the deficiencies of the ground truth segmentation mentioned above.

PPE's high accuracy comes at a price: On average, processing a $500 \times 500$ scan using our open-source implementation takes $1.6\,\mathrm{h}$ on a single core of an Intel Xeon CPU with $2.6\,\mathrm{GHz}$, while our MATLAB implementation of MSAC needs $1.1\,\mathrm{s}$. As a method specifically developed to enable real-time plane extraction, PEAC runs at approximately $30\,\mathrm{Hz}$ on an Intel i7-7700K processor.

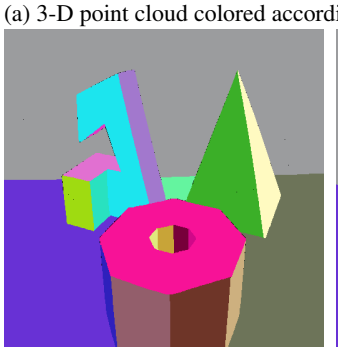
## V. Conclusion and Future Work

Many authors have investigated the problem of extracting planes from 3-D laser scans and proposed solutions. The present paper sets itself apart in two ways. First, it proposes PPE, an approach to plane extraction that builds upon an accurate probabilistic sensor model instead of the conventional point-to-plane distance heuristic. Our experiments demonstrate that the accuracy of the sensor model translates to superior plane reconstruction results. Second, motivated by the deficiencies of the popular plane extraction benchmark SegComp, we suggest an alternative benchmark, dubbed SynPEB. Both the implementation of the proposed algorithm and the suggested dataset are available online [1].

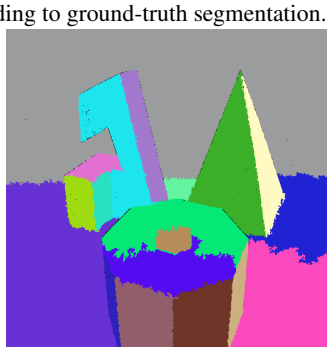
Due to the promising results, we plan several extensions of PPE. First of all, we will decrease the runtime to enable online plane extraction. In addition, we will relax the requirement that the point cloud is organized, and investigate whether leveraging laser remission intensity information can further improve the results.



(a) 3-D point cloud colored according to ground-truth segmentation.



(b) Ground-truth segmentation.



(c) PPE segmentation.

Fig. 3: Point cloud and segmentation images of scan `perc.test.23` of the SegComp dataset. Outliers are colored black.

## REFERENCES


[1] A. Schaefer, J. Vertens, D. Büscher, and W. Burgard. (2019) Probabilistic Plane Extraction. [Online]. Available: https://github.com/acschaefer/ppe

[2] A. Schaefer, D. Büscher, L. Luft, and W. Burgard, "A maximum likelihood approach to extract polylines from 2-D laser range scans," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018.

[3] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher, "An experimental comparison of range image segmentation algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 673–689, July 1996.

[4] D. Hähnel, W. Burgard, and S. Thrun, "Learning compact 3D models of indoor and outdoor environments with a mobile robot," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 15–27, 2003.

[5] J.-E. Deschaud and F. Goulette, "A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing," in *Proceedings of the 5th International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2010.

[6] A. Nurunnabi, D. Belton, and G. West, "Robust segmentation in laser scanning 3D point cloud data," in *International Conference on Digital Image Computing Techniques and Applications (DICTA)*, December 2012, pp. 1–8.

[7] Z. Dong, B. Yang, P. Hu, and S. Scherer, "An efficient global energy optimization approach for robust 3D plane segmentation of point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 137, pp. 112–133, 2018.

[8] D. Holz and S. Behnke, "Fast range image segmentation and smoothing using approximate surface reconstruction and region growing," in *Intelligent Autonomous Systems 12*. Springer, 2013, pp. 61–73.

[9] P. F. Proença and Y. Gao, "Fast cylinder and plane extraction from depth cameras for visual odometry," *Computing Research Repository (CoRR)*, vol. abs/1803.02380, 2018.

[10] X. Jiang and H. Bunke, "Fast segmentation of range images into planar regions by scan line grouping," *Machine Vision and Applications*, vol. 7, no. 2, pp. 115–122, June 1994.

[11] C. Cabo, S. G. Cortés, and C. Ordoñez, "Mobile Laser Scanner data for automatic surface detection based on line arrangement," *Automation in Construction*, vol. 58, pp. 28–37, 2015.

[12] R. Hoffman and A. K. Jain, "Segmentation and classification of range images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 608–620, September 1987.

[13] A. J. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen, "Efficient organized point cloud segmentation with connected components," *Semantic Perception Mapping and Exploration*, 2013.

[14] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6218–6225.

[15] R. T. Marriott, A. Pashevich, and R. Horaud, "Plane-extraction from depth-data using a Gaussian mixture regression model," *Computing Research Repository (CoRR)*, vol. abs/1710.01925, 2017.

[16] T. T. Pham, M. Eich, I. Reid, and G. Wyeth, "Geometrically consistent plane extraction for dense indoor 3D maps segmentation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016, pp. 4199–4204.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[18] P. F. U. Gotardo, O. R. P. Bellon, and L. Silva, "Range image segmentation by surface extraction using an improved robust estimator," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, June 2003, pp. II–33.

[19] O. Gallo, R. Manduchi, and A. Rafii, "CC-RANSAC: fitting planes in the presence of multiple surfaces in range data," *Pattern Recognition Letters*, vol. 32, no. 3, pp. 403–410, 2011.

[20] A. Sveier, A. L. Kleppe, L. Tingelstad, and O. Egeland, "Object detection in point clouds using conformal geometric algebra," *Advances in Applied Clifford Algebras*, vol. 27, no. 3, pp. 1961–1976, September 2017.

[21] M. Alehdaghi, M. A. Esfahani, and A. Harati, "Parallel RANSAC: speeding up plane extraction in RGBD image sequences using GPU," in *International Conference on Computer and Knowledge Engineering*, October 2015, pp. 295–300.

[22] G. Vosselman, B. Gorte, G. Sithole, and T. Rabbani, "Recognising structure in laser scanning point clouds," in *Proceedings of the ISPRS working group VIII/2: laser scanning for forest and landscape assessment*, M. Thies, B. Koch, H. Spiecker, and H. Weinacher, Eds. University of Freiburg, October 2004, pp. 33–38.

[23] B. Oehler, J. Stueckler, J. Welle, D. Schulz, and S. Behnke, "Efficient multi-resolution plane segmentation of 3D point clouds," in *International Conference on Intelligent Robotics and Applications*. Springer, 2011, pp. 145–156.

[24] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter, "The 3D Hough transform for plane detection in point clouds: a review and a new accumulator design," *3D Research*, vol. 2, no. 2, p. 3, November 2011.

[25] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected papers of Hirotugu Akaike*. Springer, 1998, pp. 199–213.

[26] Mitsubishi Electric Research Laboratories. (2018) PEAC. [Online]. Available: http://www.merl.com/research/?research=license-request&sw=PEAC

[27] P. H. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer vision and image understanding*, vol. 78, no. 1, pp. 138–156, 2000.