Highly Parallelizable Plane Extraction for Organized Point Clouds Using Spherical Convex Hulls

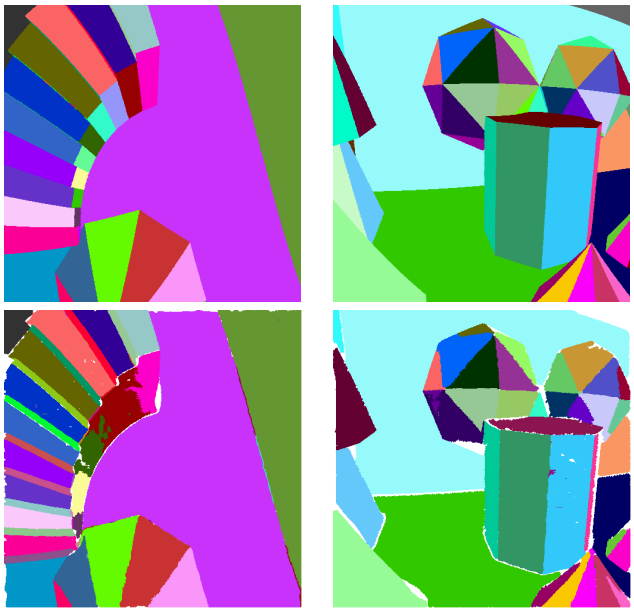
Hannes Möls, Kailai Li, and Uwe D. Hanebeck

***Abstract*— We present a novel region growing algorithm for plane extraction of organized point clouds using the spherical convex hull. Instead of explicit plane parameterization, our approach interprets potential underlying planes as a series of geometric constraints on the sphere that are refined during region growing. Unlike existing schemes relying on downsampling for sequential execution in real time, our approach enables pixelwise plane extraction that is highly parallelizable. We further test the proposed approach with a fully parallel implementation on a GPU. Evaluation based on public data sets has shown state-of-the-art extraction accuracy and superior speed compared to existing approaches, while guaranteeing real-time processing at full input resolution of a typical RGB-D camera.**

I. Introduction

Extracting high-level geometric information plays a critical role for robotic perception tasks, especially in man-made environments mostly composed of planar structures, e.g., walls, streets, ceilings and floors. Compared to pointwise feature extraction, planes can provide more reliable and predominant primitives. As an underlying map representation, they also significantly reduce memory consumption and runtime for further processing. Therefore, plane extraction has become increasingly popular in various application scenarios such as simultaneous localization and mapping (SLAM), human-machine interaction, sensor calibration, robotic locomotion, as well as virtual/augmented reality [1]–[8], etc.

More specifically, plane extraction techniques are expected to give pixelwise associations to planes clustered from point clouds. Multiple sensory modalities can hereby be involved, e.g., stereo cameras, laser scanners, light detection and ranging (LiDAR) sensors, RGB-D(epth) cameras or any multi-sensor setups able to stream out three-dimensional pointwise geometric information. The popularization of consumer-affordable 3D sensing solutions endows mobile platforms profound elevation of perception capabilities, providing image-level resolution of point cloud readings. The point cloud data is usually organized in an image-like structure (so-called organized point cloud). However, large amounts of data need to be processed using typical embedded computers, ideally without considerable sacrifice of extraction precision and level of detail. Some hardware for parallel computing (e.g., embedded GPUs) exists [9], however, parallelizing existing sequential plane extraction algorithms is non-trivial. Given inputs of organized point clouds, existing plane extraction techniques can be distinguished into three categories, namely, approaches based on (1) random sample consensus (RANSAC), (2) the Hough transform, and (3) region growing.



(a) (b)

Fig. 1: Plane extraction results (bottom) using the proposed approach. Two example frames (ground truth shown on top) were taken from the SynPEB [10]. Execution time is around 15 ms per frame for a full resolution of 500×500 pixels.

RANSAC is a model-fitting algorithm specifically designed to overcome problems introduced by strong noise and outliers. To get planes extracted, it iteratively splits a given set of input points into disjoint inlier and outlier sets. The inliers are then used to calculate the plane equation from three points. The process of randomly choosing the inlier set is repeated until a certain metric (e.g., point-to-plane distances) is valued as sufficiently optimal. Schabel et al. [11] used RANSAC to approximate point clouds with primitive geometric shapes using an octree subdivision of the scene. However, in order to find a good inlier set, a large number of potential inlier/outlier splits have to be tested which is very time-consuming. Alehdagi et al. [12] parallelized RANSAC on the GPU. While achieving a significant performance boost,

The work is supported by German Research Foundation (DFG) under grant HA 3789/16-1. The authors are with the Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), Germany. E-mails: ureaa@student.kit.edu, kailai.li@kit.edu, uwe.hanebeck@kit.edu.

the algorithm is still not capable to run in real time for full-resolution input (< 10 Hz given 640×480 depth image) due to the inherent redundancy of RANSAC trials.

Hough transform was originally designed for image processing and extracts geometric primitives, e.g., circles and lines, by voting in a discretized parameter space. A first extension for three-dimensional plane extraction was proposed in [13]. Planes are computed for each input point and fed through the voting procedure in the parameter space. Due to memory constraints, similar planes are mapped to the same cell in the parameter space which is memory-inefficient. Though mapping and voting can be trivially parallelized, discretizing a large parameter space is rather memory-consuming and cannot be tailored for faster and smaller memory. This prohibits exploiting the caching mechanism of modern GPUs. Moreover, the extraction can suffer from loss of details due to the discretization. Oehler et al. [14] proposed a Hough-RANSAC hybrid in a coarse-to-fine scheme for better trade-off between speed and extraction detailedness. Following a pre-segmentation using the Hough transform in the coarse level, RANSAC is further employed as post-processing for fleshing out the details. Unluckily, this approach is still not real-time capable for pixelwise extraction. For instance, 2.06 sec is needed for processing a 640×480 depth image from a typical RGB-D camera.

Region Growing-based extraction approaches are motivated by the grid nature of organized point clouds. Though faster than their RANSAC and Hough counterparts, region growing-based schemes cannot be further accelerated due to the exceptional difficulty of performing parallelization. The core concept of region growing-based extraction is to start at a set of seed points in a depth/normal map and to find all connecting neighbors sharing the same plane. Different metrics are used to determine if a connected neighboring pixel belongs to the same plane or not. Poppinga et al. [15] essentially applied incremental principal component analysis (PCA) to find the plane equation which best-fits all points and used the incremental mean squared error (MSE) to cluster newly observed points. Holz et al. [16] followed the same concept but utilized normal-based region growing on meshes instead. While much acceleration potential is present thanks to partial parallelization, it is error-prone for growing planes on large low-curvature surfaces since it uses a normal centroid to update the plane estimate. In [17], PCAs are applied for getting patchwise plane equations. Agglomerative clustering based on MSE are then performed to join patches to larger planes. Therefore, only planes larger than the patch size can be found and extra post-processing is needed for sub-patch segmentation. Full parallelization of the core algorithm is almost infeasible, since it uses a min-heap which changes in each iteration. Schäfer et al. [10] employed maximum likelihood estimation (MLE) to maximize the likelihood of the laser scan measurement in a similar agglomerative hierarchy. Though it can generate very accurate results despite large noise, the algorithm is time-consuming and notably difficult to parallelize due to similar limitations mentioned before. In [18], a region growing approach was proposed in a discretized space of $\mathbb{R}^3$ for the normals. However, it also suffers from loss of details due to discretization while being dependent on a complex look-back strategy for the clustering.

The state-of-the-art algorithms for plane extraction predominantly rely on explicit parameterization of the planes and solving optimization problems is often involved. With complex pipeline designs for multi-resolution schemes, some extraction algorithms can be accelerated. However, sacrifices on the extraction details are inevitable. In fact, for full-resolution plane extraction, the execution time of existing approaches ranges from several hundred milliseconds to hours, which is not real-time capable. Therefore, they are less appealing for robotic applications where both real-time performance and perception detailedness are desired within constraints of embedded computational resources. The aforementioned issues thus strongly motivate the development of a lightweight algorithm that is inherently parallelizable for enabling fast and detailed plane extraction for 3D robotic vision.

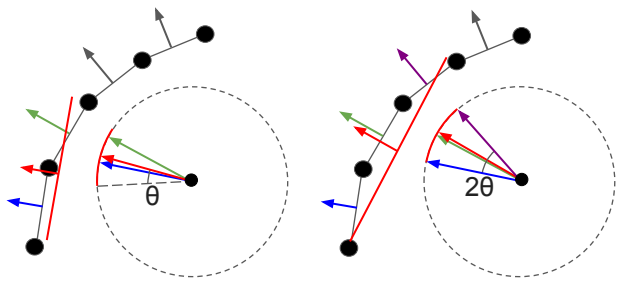
In this paper, we introduce a non-heuristic approach for extracting planes from organized point clouds in a highly parallelizable manner. In contrast to the approaches mentioned above, our major contributions are highlighted in two aspects. First, plane segments are not explicitly parameterized but interpreted as a series of geometric constraints on the spherical domain. Second, region growing in the normal map is regulated by the convex hull associated to the plane on the sphere, which inherently guarantees parallel processing. The algorithm is light-weight and enables pixelwise plane extraction for low-cost GPU in real time. We compare our system with other popular plane extraction schemes based on publicly available data sets SegComp [19] and SynPEB [10]. To the best knowledge of the authors, our algorithm provides the fastest plane extraction by far.

The remaining part of the paper is structured as follows. In Sec. II, preliminaries about convex hull and geometric interpretation for plane extraction from normals are given. The detailed algorithm for spherical convex hull-based region growing (SCH-RG) and the extraction pipeline is introduced in Sec. III. Based thereon, the implementation details are given in Sec. IV and a thorough evaluation using public data sets is performed in Sec. V. Finally, the work is concluded in Sec. VI.

II. Preliminaries

A. Convex Hull on the Sphere

The convex hull of a finite point set is defined as the convex combination of the points. When generalizing this concept to the spherical domain, a straight line between two points then adapts itself to be the shortest arc on $\mathbb{S}^2$. Given a set of normals on the sphere $\mathbb{P} \subset \mathbb{S}^2$, its convex hull $\mathrm{Conv}(\mathbb{P})$ is the smallest spherical polygon covering all the points of $\mathbb{P}$. Points on the edge cycle of the polygon are called vertices, forming the set $\mathbb{V}$. The vertices surround the point set $\mathbb{U}$ inside the polygon, namely, $\mathrm{Conv}(\mathbb{P}) = \mathbb{U} \cup \mathbb{V}$ and $\mathbb{U} \cap \mathbb{V} = \emptyset$.



(a) (b)

Fig. 2: Plane extraction on the circular domain. Normals (arrows) between point cloud vertices (dots) are located on $\mathbb{S}^1$ (dashed).

B. Plane Extraction Condition

When growing clusters for normals, a certain metric is needed to threshold newly received normal. Considering the manifold structure of the spherical domain, the arc length is hereby employed. For a set of normals $\mathbb{P} = \{\mathbf{p}_i\} \subset \mathbb{S}^2$, an underlying plane can be extracted if there exists a plane normal $\boldsymbol{\mu}$ within the range of angle θ to all normals, namely

$$\{\, \boldsymbol{\mu} \in \mathbb{S}^2 \,|\, \angle(\boldsymbol{\mu}, \mathbf{p}_i) \leq \theta, \forall\, \mathbf{p}_i \in \mathbb{P} \,\} \neq \emptyset\,. \tag{1}$$

C. Inspiration of Plane Extraction on Circular Domain

Based on the aforementioned condition, a plane can be extracted by finding its normal satisfying the condition in (1), meanwhile maximizing the cardinality $|\mathbb{P}|$ of the clustered normals. Before introducing the proposed algorithm on the spherical domain, we first illustrate the intuition behind it by taking an example of clustering normals on the circular domain. As shown in the left side of Fig. 2, we start from the initial normal in blue. When considering the next connected normal (green), the condition in (1) should hold, meaning that there exists a red arc segment of length 2θ covering both normals. However, a constrained yet infinite amount of arc segments can be found at this stage. Therefore, the red plane normal $\boldsymbol{\mu}$ is not fixed. On the right side of Fig. 2, the next neighboring normal (purple) is added and is assumed to fully constrain the location of the red arc segment, fixing the plane normal. It should be noted that only the bordering pair of normals constrain the plane normal $\boldsymbol{\mu}$ on the circular domain.

This intuition can be generalized to the spherical domain for plane extraction by extending the bordering pair on the circle to the spherical convex hull. When considerably many vertex normals are added, the algorithm shows asymptotic behavior of approximating the cluster boundary (even for slightly curved surfaces).

III. Proposed Plane Extraction Approach

A. Spherical Convex Hull-based Region Growing (SCH-RG)

As concluded in Sec. II-C, the plane normal $\boldsymbol{\mu}$ is determined by the spherical convex hull (SCH) of normals satisfying the extraction condition in (1). We introduce the so-called *cluster permissible region* (CPR) (shown as orange line segments in Fig. 3). It is an approximation of the set of potentially valid normals satisfying the condition in (1) given the current normal set $\mathbb{P} = \{\mathbf{p}_i\}$. For one single normal $\mathbf{p}_i$, its CPR is the set of neighboring normals within the radius of 2θ, i.e.,

$$\mathrm{CPR}\,(\mathbf{p}_i) = \{\, \mathbf{q} \in \mathbb{S}^2 \,|\, \angle(\mathbf{p}_i, \mathbf{q}) \leq 2\theta \,\}\,. \tag{2}$$

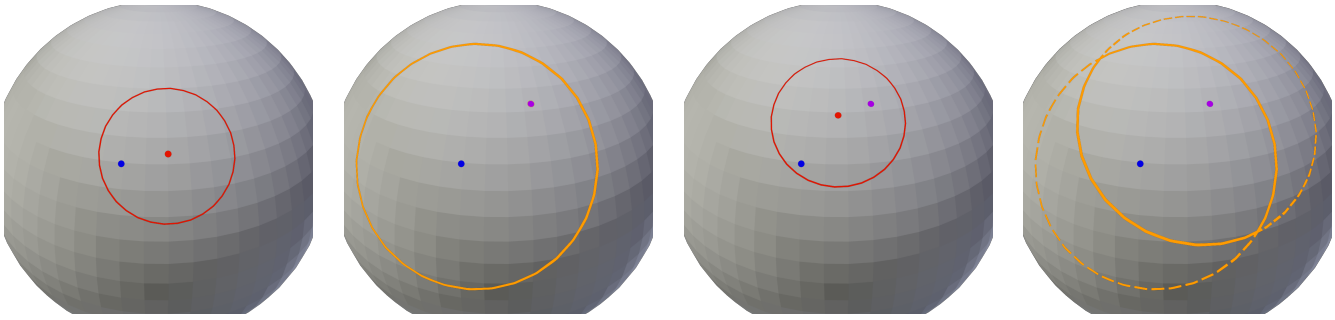
The CPR of multiple normals is then the intersection of CPRs of individual normals, namely $\mathrm{CPR}\,(\mathbb{P}) = \bigcap_{\mathbf{p}_i \in \mathbb{P}} \mathrm{CPR}(\mathbf{p}_i)$.

There are two important properties of the CPR that are exploited: (i) If the newly received normal is covered by the current CPR, a valid plane normal $\boldsymbol{\mu}$ is assumed to exist and the CPR is then updated via intersection (see Fig. 3.5 – 3.7). (ii) The CPR of a point set $\mathbb{P}$ is only determined by the intersection of CPRs of the SCH vertices $\mathbb{V}$, i.e., $\mathrm{CPR}(\mathbb{P}) = \mathrm{CPR}(\mathbb{V})$. CPRs of the normal set $\mathbb{U}$ that are inside $\mathrm{Conv}(\mathbb{P})$ are therefore redundant for performing intersection (see Fig. 3.9). Therefore, the shape of the CPR varies only in the case of a changing convex hull and the spherical convex hull is always a subset of the CPR, i.e., $\mathrm{Conv}(\mathbb{P}) \subset \mathrm{CPR}(\mathbb{P}) = \mathrm{CPR}(\mathbb{V})$. Also, we have $\mathrm{CPR}(\emptyset) = \emptyset$. As the CPR is a superset of the set of all valid normals, property (i) can be violated, resulting in the covering of outlier normals in theory. In practice, however, this side effect occurs extremely rarely which brings no degeneration without explicit handling.

Algorithm 1: SCH-based Region Growing (SCH-RG)

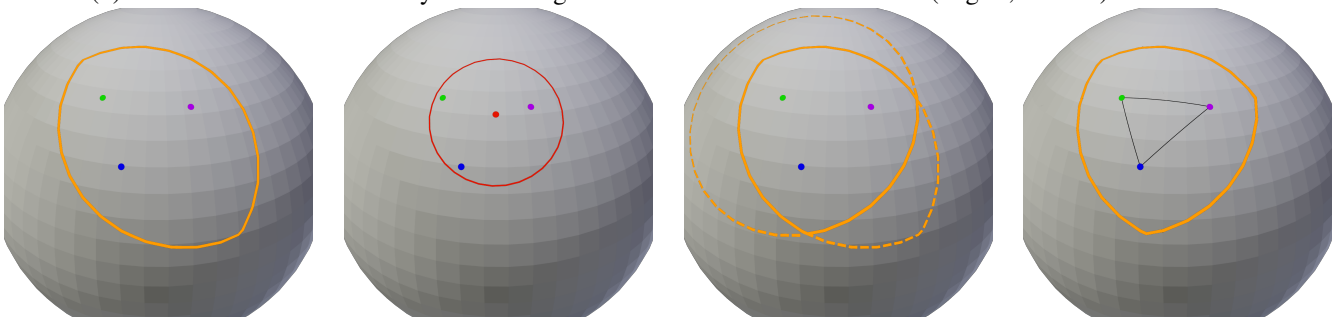
Data: seed $\mathbf{s}$, already clustered point set $\mathbb{T}$
Result: newly clustered point set $\mathbb{P}$

$\mathbb{P} \leftarrow \{\mathbf{s}\}$
$(\mathbb{U}, \mathbb{V}) \leftarrow \mathrm{InitSCH}\,(\mathbb{P})$
$\mathbb{Q} \leftarrow \mathrm{QueueUpNeighbors}\,(\mathbf{s})$
while $\mathbb{Q} \neq \emptyset$ **do**
  $\mathbf{p} \leftarrow \mathrm{Pop}(\mathbb{Q})$
  $d_p \leftarrow \mathrm{GetDepth}(\mathbf{p})$
  $d_o \leftarrow \mathrm{GetOriginDepth}(\mathbf{p})$
  if $\mathbf{p} \in \mathbb{P}$ *or* $\mathbf{p} \in \mathbb{T}$ *or* $|d_o - d_p| > T$ **then**
    continue `// Discard p`
  end
  if $\mathbf{p} \in \mathbb{U}$ **then**
    $\mathbb{P} \leftarrow \mathbb{P} \cup \mathbf{p}$
    $\mathbb{Q} \leftarrow \mathrm{QueueUpNeighbors}(\mathbf{p})$
  else
    if $\mathbf{p} \notin \mathit{CPR}(\mathbb{V})$ **then**
      continue `// Discard p`
    else
      if $|\mathbb{V}| < 3$ **then**
        $\mathbb{V} \leftarrow \mathbb{V} \cup \mathbf{p}$
      else
        $\mathbb{V} \leftarrow \mathrm{ExpandSCH}(\mathbf{p})$
      end
      $\mathbb{P} \leftarrow \mathbb{P} \cup \mathbf{p}$
      $\mathbb{Q} \leftarrow \mathrm{QueueUpNeighbors}(\mathbf{p})$
    end
  end
end



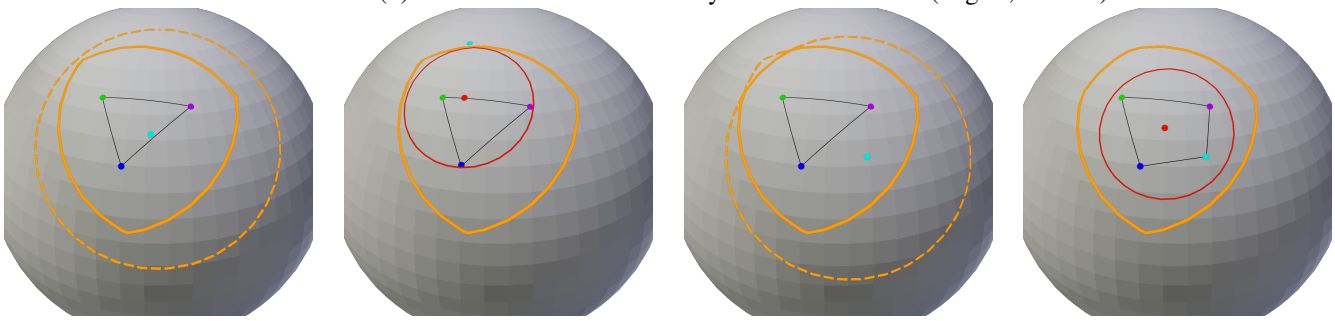
(1) (2) (3) (4)

(1) A single normal (blue) and a possible $\boldsymbol{\mu}$ containing the plane normal (Alg. 1, line 1-2). (2) A second normal (violet) is covered by the CPR of the blue normal (Alg. 1, line 17). (3) Because the violet normal is within the CPR, a plane normal $\boldsymbol{\mu}$ containing both plane normals can be found. There are many plane normals satisfying condition in (1) while containing both normals. (4) The new CPR is created by intersecting individual CPRs of the normals (Alg. 1, line 19).



(5) (6) (7) (8)

(5) A third normal (green) lies within the CPR of the blue and violet normals (Alg. 1, line 17). (6) A potential plane normal $\boldsymbol{\mu}$ exists, which contains all three normals. Note that $\boldsymbol{\mu}$ is now more constrained than in (3). (7) The new CPR of all three normals is created via intersection. (8) A convex hull is formed by the three normals (Alg. 1, line 19).



(9) (10) (11) (12)

(9) A new normal (cyan) was added and lies within the convex hull, therefore it is part of the current plane but does not update the convex hull (Alg. 1, line 12). (10) A new normal (cyan) lies outside of the convex hull and outside of the CPR, therefore no plane normal can be found for all of the four normals. It is thus not added to the current plane (Alg. 1, line 16). (11) A new normal (cyan) lies outside of the convex hull but inside the CPR (Alg. 1, line 17). (12) The convex hull is updated (Alg. 1, line 21) and shrinks the CPR via intersection. Note that valid plane normal set is constrained even more.

Fig. 3: Illustration of the proposed plane extraction algorithm based on spherical convex hulls.

The proposed approach for finding one single plane from one given seed point is illustrated in Fig. 3 and introduced in detail in Alg. 1. As the algorithm runs in parallel, a set $\mathbb{T}$ of already clustered points is given as input. For initializing seed points, an efficient algorithm to find seeds for all underlying planes in a frame is introduced later in III-B. The proposed algorithm is to be executed for each of these initialized seed points. A first-in-first-out (FIFO) queue $\mathbb{Q}$ is applied for checking whether a pixel normal is to be clustered into the same plane. It should be noted that points from a parallel offset plane cannot be directly recognized by only region growing on the normal map. To solve this issue, a depth check between neighboring pixels is performed alongside the region growing. The connected neighboring pixels in

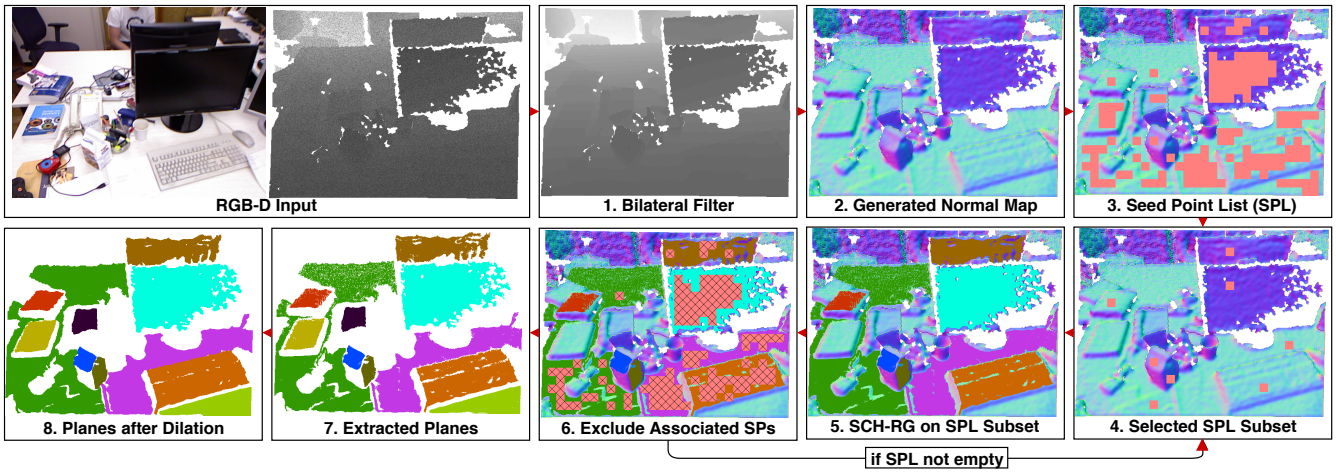


Fig. 4: Overview of the proposed plane extraction approach based on region growing using spherical convex hulls.

the queue also carry the depth value of its origin pixel. If the depth difference is larger than a predefined threshold T (Alg. 1, line 6-8), the connected pixel is discarded from the plane of the seed pixel. Theoretically, this can falsely discard connected neighboring pixels on planes lying steeply relative to the camera angle. In practice, however, little side effects are observed. Alternative approaches for solving this issue are, e.g., to incorporate color intensity for validation, or to apply on-plane validation for the neighboring pixels based on the point-to-plane metric. Here, explicit plane parameterizations are essential, which prohibit the extraction parallelizability.

B. System Overview

Based on the proposed region growing using spherical convex hulls, we introduce a light-weight plane extraction approach shown in Fig. 4. Given a depth image, e.g., streaming out from the RGB-D camera [20], a bilateral filter is first applied for denoising, after which the normal map is generated (shown in Fig. 4.1 – 4.2). A seed point list (SPL) is constructed in a patch-wise fashion (e.g., of size 20×20 pixels, shown as red boxes in Fig. 4.3). Here, the MSE of point-to-plane fits is computed for each patch [16]. A given MSE threshold is then applied to discard patches on plane transitions and curved surfaces. The remaining patches are then sorted according to their patch-plane roughness and their center points are gathered into the seed point list (SPL) accordingly.

Each time a subset is selected from the SPL such that at most one seed is placed on each unextracted plane (Fig. 4.4). Afterwards, the region growing algorithm is performed using SCH-RG (Fig. 4.5). It should be noted that some valid seed points in the SPL are associated to extracted planes during the current region growing. In this case, they are removed from the SPL (Fig. 4.6). The procedure of region growing and seed exclusion are repeated until no remaining normals can be found in the SPL. As post-processing, dilation is executed on the extracted planes to eliminate holes caused by the noise for practical applications.

The usage of the subset mechanism is to avoid merging clusters and to ensure that one plane can be cleanly flooded over from one single seed point. To avoid rivalry cases where two seed points grow regions on the same plane (oversegmentation), we select seeds with angle differences of patch plane normals larger than 2θ. Seed points that are falsely excluded here can still be processed in the subsets later. Our tests have shown that enough seed points can be selected to fully occupy the GPU.

IV. Implementation

The proposed region growing algorithm is computationally intensive, but inherently parallelizable. More specifically, its parallelizability can be interpreted in three execution levels: (i) At the pixel level, validating the CPR and convex hull for a single pixel normal can be easily parallelized. (ii) At the plane level, parallel validation of normals in queue $\mathbb{Q}$ (i.e., the while-loop in Alg. 1) w.r.t. to the current CPR is guaranteed. (iii) At the seed/cluster level, multiple planes can be flooded from different seeds in parallel.

The proposed plane extraction approach is prototyped in CUDA and C++ with a NVIDIA MX150 GPU. Every step of the system workflow in Fig. 4 is fully parallelized. Steps in the loop shown in Fig. 4.4 – 4.6 also run in parallel to each other (e.g., new SPL subset can be selected while the region growing is performed). In practice, the proposed algorithm is executed in a CUDA cooperative kernel. It is preferable to fully occupy the workers on the GPU and use a software block-wise scheduler for achieving good performance. The block-wise execution gives the advantage of using the fast shared memory. Within each block, pixels are processed warp-wise. This is a necessity because mutual exclusions, which are needed when expanding the convex hull, only work at warp level. Thread-wise execution is hereby avoided since threads within the same warp on different execution paths are processed sequentially, which gives worse performance.

If a newly received normal lies within the CPR, its arc-length distances to all the vertices of the convex hull need to be smaller than 2θ (described in Sec. III-A). This validation is done on 32 threads in parallel. We store the spherical convex hull as a set of normals given by planes intersecting the $\mathbb{S}^2$ center and convex hull edges. The normals are oriented

Approach	f in %	α	n_o	n_u	n_m	n_s
SegComp Perceptron [19]						
USF [19]	60.9	2.7	0.4	0.0	5.3	3.6
WSU [19]	40.4	3.3	0.5	0.6	6.7	4.8
UB [19]	65.7	3.1	0.6	0.1	4.2	2.8
UE [19]	68.4	2.6	0.2	0.3	3.8	2.1
UFPR [21]	75.3	2.5	0.3	0.1	3.0	2.5
Oehler et al. [14]	50.1	5.2	0.3	0.4	6.2	3.9
Holz et. al. [16]	75.3	2.6	0.4	0.2	2.7	0.3
RPL-GMR [10]	72.4	2.5	0.3	0.3	3.0	2.0
Feng et al. [17]	60.9	2.4	0.2	0.2	5.1	2.1
PEAC [10]	48.6	2.6	0.0	0.1	7.1	2.0
MSAC [22]	18.5	3.9	0.1	0.2	11.3	3.4
PPE [10]	60.7	2.8	1.4	1.1	1.5	2.3
SCH-RG (proposed)	73.4	4.0	1.5	0.5	5.8	7.0
SynPEB [10]						
PEAC [10]	29.1	–	0.7	1.0	26.7	7.4
MSAC [22]	7.3	–	0.3	1.0	36.3	10.9
PPE [10]	73.6	–	1.5	1.1	7.1	16.5
SCH-RG (proposed)	58.6	–	2.4	1.0	24.1	17.8

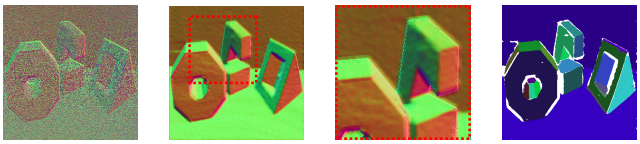
TABLE I: Accuracy comparison of the proposed algorithm SCH-RG with existing approaches. f in % is the percentage of correctly segmented planes. α is the angular std. deviation w.r.t the ground truth. n_o, n_u, n_m, n_s are the average amount of oversegmented , undersegmented, missing, and spurious planes per frame. Noise level for SynPEB is 1 mdeg.

towards the hull center. If a newly received normal belongs to the current spherical convex hull, its distances to all the hull planes should be positive. This test is also executed thread-wise in parallel using the 32 warp threads.

V. Evaluation

Evaluating the proposed plane extraction method on the well-known data sets SegComp [19] and SynPEB [10] has given state-of-the-art accuracy (shown in Tab. I). The metric definition and evaluation results are copied from [10]. The proposed method provides superior extraction speed with an improvement of at least one order of magnitude compared to PEAC [17], the fastest plane extraction algorithm in the table. PEAC runs 120 ms per frame for a 640×480 depth image using patches of 4×4 pixels. For the same image size, our proposed SCH-RG runs 17 ms per frame pixelwise on a NVIDIA MX150 GPU. This can be broken down to 2.2 ms for the bilateral filter and normal map generation, 0.66 ms to find and sort the image patches, 0.37 ms to extract the first subset and 13.9 ms for the core SCH-RG as well as 0.2 ms for dilation. Consecutive subset extractions are done in parallel to the region growing. Because SCH-RG is a region growing algorithm, the execution time is proportional to the amount of pixels. One can argue that the improvement merely benefits from running the algorithm on the GPU. However, most of the alternative algorithms are hardly, or not at all, parallelizable. For instance, PEAC uses a min-heap to get the next element to be processed. Once processed, the min-heap changes accordingly. This means that the core execution step can only be processed sequentially.

The most theoretically similar algorithm to SCH-RG is the one from Holz et al. [16] as it uses region growing on a bilateral-filtered normal map. However, its metric for adding new normals to a given plane requires averaging all



(a) raw (b) filtered (c) artifact (d) result

Fig. 5: Example image from SegComp for showing the artifact due to the bilateral filter.

normals currently covered by the plane. For low-curvature surfaces, the average normal will be slowly dragged along the curvature during region growing, thereby falsely classifying these surfaces as planes. Furthermore, already clustered normals can "fall out" of a plane if a majority of the newly clustered normals have dragged the average normal too far away. The authors tried to curb this problem by adding an additional point-to-plane distance check. SCH-RG does not suffer from this issue as it does not rely on averaging normals. Furthermore, [16] can only be executed sequentially since a new average is calculated for each addition of a new normal. In contrast, it is observed in the proposed approach that there are 99.35% of normals which fall inside the convex hull and do not contribute to its shaping. In our test, e.g., the convex hull of a large plane with 20k+ pixels is only expanded 130 times during region growing with around 40 vertices.

In the proposed approach, the seed patch size neither affects the extraction speed nor the accuracy. In each seed patch, only a single pixel is used during region growing, regardless of the patch size. The region growing itself is always executed pixelwise. The seed patch size merely determines the size of extracted planes and introduces a slight overhead during the extraction and sorting of seed points. Furthermore, directly applying the proposed algorithm on raw and noisy input requires a large θ, leading to undersegmentations. For denoising purposes, the bilateral filter is employed. However, this also creates artifacts as shown in Fig. 5-(c). The "glowing" of the edges has to be accepted and is believed to be the major reason of the accuracy depredation compared to the most accurate algorithms.

VI. Conclusion

In this paper, a novel region growing-based plane extraction technique is proposed for organized point clouds using spherical convex hulls. It gives state-of-the-art extraction accuracy and is inherently designed for parallel computing, thus achieving the fastest pixelwise processing rate among all existing approaches even on low-cost GPUs.

Though promising results have been shown in this work, much potential can still be exploited. For instance, directional statistics and estimation approaches [23], [24] can be considered for further handling the uncertainty on the sphere. As plane representations are much more memory-efficient, the proposed parallel plane extraction can be integrated into SLAM frameworks for real-time large-scale reconstruction and long-term navigation for autonomous robots.

REFERENCES

[1] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and Y. Miki, "Calibration of Non-overlapping Cameras Using an External SLAM System," in *Proceedings of the 2014 IEEE International Conference on 3D Vision (3DV 2014)*, Dec. 2014, pp. 509–516.

[2] K. Li, F. Pfaff, and U. D. Hanebeck, "Geometry-driven stochastic modeling of se(3) states based on dual quaternion representation," in *Proceedings of the 2019 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2019)*, Taipei, Republic of China, May 2019.

[3] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA 2016)*, 2016, pp. 1285–1291.

[4] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "SLAM Using Both Points and Planes for Hand-Held 3D Sensors," in *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2012)*, Nov. 2012.

[5] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini, "Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality," *Springer Tracts in Advanced Robotics (STAR)*, 2019.

[6] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-Plane SLAM for Hand-Held 3D Sensors," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA 2013)*, Karlsruhe, Germany, May 2013, pp. 5182–5189.

[7] Y. Shi, K. Xu, M. Nießner, S. Rusinkiewicz, and T. Funkhouser, "Planematch: Patch coplanarity prediction for robust rgb-d reconstruction," in *Proceedings of the 2018 European Conference on Computer Vision (ECCV 2018)*, 2018, pp. 750–766.

[8] S. Bultmann, K. Li, and U. D. Hanebeck, "Stereo visual slam based on unscented dual quaternion filtering," in *Proceedings of the 22nd International Conference on Information Fusion (Fusion 2019)*, Ottawa, Canada, July 2019.

[9] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-Time Large-Scale Dense 3D Reconstruction with Loop Closure," in *Proceedings of the 2016 European Conference on Computer Vision (ECCV 2016)*, 2016, pp. 500–516.

[10] A. Schaefer, J. Vertens, D. Büscher, and W. Burgard, "A Maximum Likelihood Approach to Extract Finite Planes from 3-D Laser Scans," in *Proceedings of the 2019 IEEE International Conference on Robotics and Automation (ICRA 2019)*, May 2019.

[11] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," in *Computer Graphics Forum*, vol. 26, no. 2. Wiley Online Library, 2007, pp. 214–226.

[12] M. Alehdaghi, M. A. Esfahani, and A. Harati, "Parallel RANSAC: Speeding Up Plane Extraction in RGBD Image Sequences Using GPU," in *Proceedings of the 2015 IEEE International Conference on Computer and Knowledge Engineering (ICCKE 2015)*, 2015, pp. 295–300.

[13] G. Vosselman, B. G. Gorte, G. Sithole, and T. Rabbani, "Recognising Structure in Laser Scanner Point Clouds," in *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 46, no. 8, 2004, pp. 33–38.

[14] B. Oehler, J. Stueckler, J. Welle, D. Schulz, and S. Behnke, "Efficient Multi-resolution Plane Segmentation of 3D Point Clouds," in *Proceedings of the 2011 International Conference on Intelligent Robotics and Application (ICIRA 2011)*. Springer, 2011, pp. 145–156.

[15] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast Plane Detection and Polygonalization in Noisy 3D Range Images," in *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, 2008, pp. 3378–3383.

[16] 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.

[17] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast Plane Extraction in Organized Point Clouds Using Agglomerative Hierarchical Clustering," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014)*, 2014, pp. 6218–6225.

[18] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-time Plane Segmentation Using RGB-D Cameras," in *Robot Soccer World Cup*. Springer, 2011, pp. 306–317.

[19] 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, 1996.

[20] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, Vilamoura, Portugal, Oct. 2012.

[21] P. F. Gotardo, O. R. P. Bellon, and L. Silva, "Range Image Segmentation by Surface Extraction Using an Improved Robust Estimator," in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, vol. 2, 2003, pp. II–33.

[22] 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.

[23] G. Kurz, I. Gilitschenski, F. Pfaff, L. Drude, U. D. Hanebeck, R. Haeb-Umbach, and R. Y. Siegwart, "Directional Statistics and Filtering Using libDirectional," *Journal of Statistical Software*, May 2019.

[24] K. Li, D. Frisch, B. Noack, and U. D. Hanebeck, "Geometry-driven deterministic sampling for nonlinear bingham filtering," in *Proceedings of the 2019 European Control Conference (ECC 2019)*, Naples, Italy, June 2019.