Real-Time LiDAR based Calculation of the Ground Plane's Normal Vector on Spherical Mobile Mapping Systems*

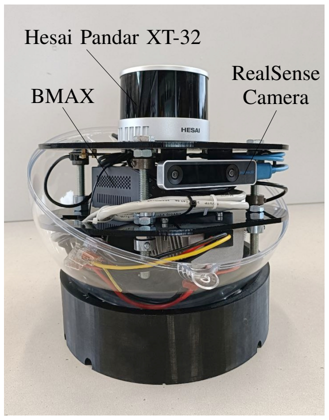
Carolin Bösch, Fabian Arzberger, and Andreas Nüchter

***Abstract*— Many mobile mapping systems utilize the on-board identification of the ground plane from 3D Light Detection and Ranging (LiDAR) data, e.g., for Simultaneous Localization and Mapping (SLAM) purposes. This enables autonomy in different environments for numerous systems such as Unmanned Aerial Vehicles (UAV) or self-driving cars. However, there is no research concerned with spherical systems on this topic. To this end, we study the real-time on-board identification of the ground plane on spherical mobile mapping systems in this work. These systems provide unique challenges such as poor ground coverage due to the weak incidence angles of the laser beams, fast rotations, and largely varying sensor orientations. We implement multiple algorithms based on point-cloud pre-processing, plane fitting, and plane detection, to address these challenges and compare them in terms of runtime, accuracy, and robustness. For this purpose we create our own datasets - including ground truth data available from a terrestrial laser scanner (TLS) - featuring different environments, slopes, and rotation speeds. Our analysis proves the real-time capability and suggests that a geometrical pre-processing approach, followed by a Random Sample Consensus (RANSAC) based plane detection algorithm outperforms the other tested approaches. Furthermore, the evaluation reveals challenges regarding the unusual locomotion mechanism and suggests that, in order to reduce the median angle error of the plane normal below 9°, we need to utilize an on-board globally consistent map, instead of only individual LiDAR frames.**

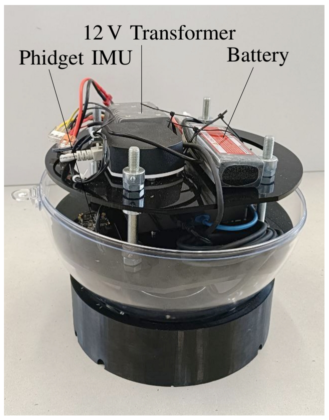
I. INTRODUCTION

The SLAM problem is one of the most relevant research fields in mobile robotics due to its implications on robot autonomy. It is often solved using perceptive and inertial sensors such as LiDAR, cameras, Inertial Measurement Units (IMUs), or a combination of them. We focus on LiDAR-inertial systems in this work. Often, the SLAM algorithms of common systems are specialized towards a specific use-case or environment. There is one specialization in particular that many SLAM algorithms use, which is the identification of the ground plane in order to restrict the optimization problem, e.g., to eliminate drift. However, many assumptions which apply to state-of-the-art systems are invalid on spherical mobile mapping systems (see Figure 1). The research field of spherical mobile mapping systems is largely under-explored, especially in terms of the SLAM problem. Yet these systems have some key advantages which make them suitable for hazardous, narrow, or inaccessible environments, such as funnels or construction sites. The spherical shell protects internal sensors from dust, water, or harsh impacts. Furthermore, the locomotion mechanism itself leads to LiDAR coverage of the whole environment without the need for additional actuators. However, spherical LiDAR-based mobile mapping systems also provide some challenges. Weak incidence angles of the laser beams hitting the ground when rolling over the floor lead to sparse coverage of the ground plane. Additionally, the unrestricted rotations which are inherent to the locomotion mechanism of a ball lead to unusual sensor orientations and thus, largely varying and sometimes obscure LiDAR readings. However, it is of great interest to have the ground plane available on-board, especially on spherical systems, because it can be directly utilized in an odometry pipeline. The cross-product of the ground plane's normal-vector and the rotation speed vector of the ball results in the linear velocity of the system. The odometry output could then be used as an input to an on-board SLAM algorithm. To this end, we explore several state-of-the-art approaches for the on-board identification of the ground plane in real-time on our spherical mobile mapping prototype. The contributions of this work are as follows:

- A ground plane identification algorithm specifically tailored towards spherical mobile mapping systems, which is able to cope with the unfavorable conditions



(a) The middle and top layer of the robot, showcasing the LiDAR, on-board computer (BMAX), and the RealSense camera.



(b) The bottom layer of the robot, featuring the power supply system, including the 12 V transformer, battery and one of the three Phidget IMUs.

Fig. 1: Overview of the spherical mobile mapping system's internal components with the open perspex shell.

*We acknowledge support from the Elite Network of Bavaria for the academic program *Satellite Technology – Advanced Space Systems*. The research was done at Julius-Maximilians-Universität Würzburg.

Carolin Bösch is with the École Nationale Supérieure de l'Aéronautique et de l'Espace. Fabian Arzberger and Andreas Nüchter are with Info XVII – Robotics at Julius-Maximilians-Universität Würzburg, Germany. Andreas Nüchter is also with the Zentrum für Telematik e.V., Würzburg and currently International Visiting Chair at U2IS, ENSTA, Institut Polytechnique de Paris, France. Contact: `fabian.arzberger@uni-wuerzburg.de`

presented by such a system. We open-source our code and datasets [1].

- A comprehensive runtime and accuracy analysis of different combinations of state-of-the-art point-cloud preprocessing and plane extraction methods. We perform tests on self-recorded datasets featuring lab- as well as outside-environments with different slopes.

II. RELATED WORK

In this section, we explore research on nearest neighbor search (NNS) libraries and multidimensional data structures for efficient point cloud processing. Further, we examine state-of-the-art plane identification methods and review existing ground-based SLAM approaches. Kd-trees are preferred in NNS libraries for LiDAR data processing due to their efficiency, simplicity, and low overhead, making them suitable for resource-limited, real-time systems [2]. In k-NNS, kd-tree-based implementations like nanoflann and FLANN offer consistent performance, low variance, and fast update times [3]. Elseberg et al. [4] found that ranged k-NNS libraries, such as 3DTK and libnabo, perform best for artificial datasets, while k-NNS (FLANN) also performs well for smaller real-world datasets.

A. Plane Identification

In general, there are two fundamental approaches to plane extraction from point clouds: plane fitting and plane detection. Whereas plane detection algorithms are able to identify subsets of points representing different planes, plane fitting algorithms blindly fit one plane to all points, often minimizing a squared point-to-plane distance residual of all the points at once. Thus, plane fitting is more susceptible to noise and outliers [5]–[7] and is only usable if we preselect a subset of points that likely represents the plane of interest. The ordinary Least Squares Fitting (LSF) and Principal Component Analysis (PCA) are the most commonly used methods for plane fitting in point clouds [6]. Ordinary LSF [8] fits a plane by minimizing the squared distances between points and the plane in one axis, while PCA [9] fits a plane considering the squared perpendicular distances. Robust PCA variants address the issues of plane fitting algorithms being susceptible to noise and outliers, but typically result in longer runtimes [10]–[16]. Comparative studies show that while PCA and LSF have similar performance, PCA often demonstrates slightly better quality in medium-sized neighborhoods and real-world data [6], [17]. However, LSF can sometimes achieve faster computation times [10]. The most commonly used plane detection algorithms are the 3D Hough Transform (HT) and RANSAC [18]. The 3D HT [19], [20] converts point clouds into a parameter space and identifies planes by finding local maxima, while RANSAC [21] iteratively fits models to subsets of data, selecting the model with the best fit. A comparative study by Tarsha-Kurdi et al. [22] finds RANSAC to be superior in terms of processing time and accuracy for detecting building roofs. Hulik et al. [23] demonstrate that RANSAC, although less stable in single-plane scenarios, often provides finer details and performs better overall than an optimized HT in varying noise conditions. Given the numerous versions of the HT, Borrmann et al. [24] demonstrate that the Randomized HT (RHT) offers significant runtime improvements over other HT methods, making it a preferred choice for quick and efficient plane detection. Nevertheless, it is difficult to make a general statement about which algorithm is superior, since their performance depends heavily on the selected parameters.

B. Ground-based SLAM

LeGO-LOAM [25] is a lightweight, ground-optimized LiDAR odometry method for real-time 6-DOF pose estimation in ground vehicles. It filters noise and extracts planar and edge features, using two-step Levenberg-Marquardt optimization to compute transformations. Compared to LOAM, it achieves similar accuracy with lower computational cost and reduces drift. An extension by Yang et al. [26] refines planar extraction using a clustering-based method, improving stability in noisy environments. Guo et al. [27] propose a 3D LiDAR SLAM method that reduces cumulative error through ground segmentation, optimizing feature extraction and global pose refinement via Scan Context loop detection. Wei et al. [28] introduce Ground-SLAM, which reduces pose drift in multi-floor environments by using LiDAR Odometry and ground planes as landmarks within pose graph optimization. Wen and Hsu [29] propose AGPC-SLAM, which improves vertical pose estimation by incorporating an absolute ground plane constraint to refine z-axis positioning and reduce drift. Liu et al. [30] present a ground plane detection method that utilizes physical constraints and flexible ground constraints to reduce vertical trajectory drift, significantly improving accuracy on both KITTI and custom datasets.

III. SPHERICAL MOBILE MAPPING SYSTEM

Our spherical mobile mapping system prototype is equipped with a clear perspex shell, as depicted in Figure 1. It is transparent to ensure that the LiDAR beams are able to effectively penetrate it. Inside the shell the robot is organized into multiple layers. The top layer houses the Hesai Pandar-XT32 [31] LiDAR which operates at 20 Hz. It features a scan range of 0.05 m to 120 m, ± 1 cm range accuracy and 0.5 cm range precision. The horizontal FOV is 360° with a resolution of 0.36° and a vertical FOV of 31° with a vertical resolution of 1°. The middle layer serves as the main platform for essential sensors and processing units, including the on-board computer, IMUs and an inside-out tracking camera. The tracking camera is not utilized in this work. The on-board computer of the spherical mobile mapping system is the BMAX MaxMini B3 Plus [32], featuring a 2.3 GHz quad-core Intel Pentium Gold 5405U processor and 8 GB of RAM. The bottom layer contains the power supply system, featuring a 12 V transformer and battery. We use the ROS for the programming of the spherical mobile mapping system's on-board computer. One program, which is specialized for spherical systems, provides an orientation estimation by filtering data from the internal IMUs [33]. Another program

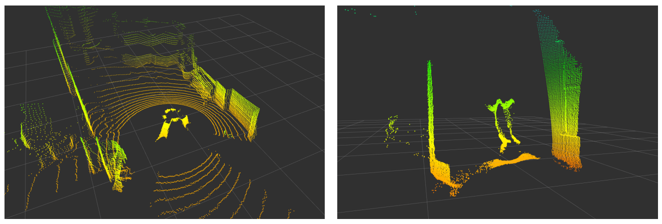

(a) LiDAR in horizontal position. (b) LiDAR in vertical position.

Fig. 2: Exemplary scans of the Hesai Pandar LiDAR during movement, illustrating that the ground remains consistently visible, and that reflections from the shell and the hands of the person who is moving the sphere are present.

performs inertial LiDAR self-motion distortion correction in real-time [34].

IV. APPROACH

In this work, we calculate the normal vector $\boldsymbol{n}$ of the ground plane in real-time using each individual LiDAR frame. The real-time requirement to calculate $\boldsymbol{n}$ within 50 ms is due to the 20 Hz frequency of the Hesai Pandar LiDAR scanner, which provides dense scans of roughly 30,000 points. It provides scans in which the ground is always visible due to its minimum scanning distance of 5 cm, refer to Figure 2. As the prototype currently has no locomotion mechanism, the scans also capture the hands of the person moving the sphere. To calculate the ground plane's normal vector, we perform the following steps on each individual LiDAR frame:

1) Pre-process the point-cloud.
2) Identify the ground plane.
3) Calculate the normal vector of the ground plane.
4) Transform the normal vector from the local LiDAR scan frame to the fixed world frame.

By implementing and comparing multiple plane identification algorithms under different pre-processing conditions, we aim to evaluate their performance regarding runtime, accuracy, and robustness in various scenarios to identify suitable solutions for spherical mobile mapping systems.

A. Ground Plane Extraction

Ground plane extraction involves algorithms designed to identify and segment the ground surface from point cloud data captured by LiDAR sensors. These plane segmentation algorithms aim to detect the dominant flat surface representing the ground. The most popular algorithms are: LSF and PCA as representatives of plane fitting methods, and RANSAC and 3D HT as examples of plane detection techniques. For this work, we implement and compare the performance of all four previously named algorithms. Furthermore, we select the RHT instead of the standard 3D HT due to its significantly faster runtime and the ball accumulator instead of the cube accumulator based on [24], [35].

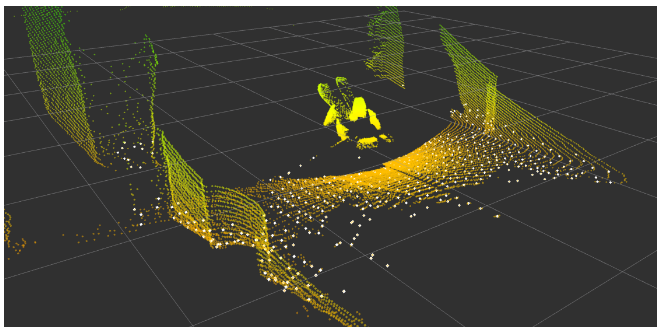


Fig. 3: Point clouds illustrating the data before (height-based color gradient) and after (white) pre-processing.

B. Pre-Processing of the Point Cloud

The plane fitting methods require pre-processing of the point cloud, since using the entire scan can lead to erroneous results due to their susceptibility to outliers. Therefore, we need to identify a subset of points, which most likely represents the ground plane. We indicate the removal of points belonging to the structure of the sphere and the hands, as well as the down-sampling of the point cloud with the keyword **filtered**. Down-sampling the point cloud reduces the number of points and achieves a more uniform point density, which is certainly desired for RANSAC and RHT. Moreover, if we remove further potential non-ground points, e.g., points belonging to walls or other objects, we refer to the result as a **sub-cloud**. Figure 3 presents an exemplary scan, illustrating the data before and after pre-processing. As pre-processing the point-cloud introduces more calculations, we need to determine the impact of these steps on runtime and accuracy, and establish a trade-off between these factors. The plane detection methods do not explicitly rely on pre-processing the point cloud. However, using the entire scan leads to multiple detected planes, requiring additional steps to identify the ground plane and increasing the runtime. Consequently, we use RANSAC and RHT to only identify the dominant plane in the sub-cloud, which we assume to be the ground plane.

1) Point Cloud Filtering: Figure 2 shows that the points belonging to the reflections of the shell, the structure of the prototype, as well as the hands of the person moving the sphere, are located within a certain radius r_{fil} around the origin of the LiDAR. To identify and remove these points, we consider the following two options:

1) **Geometrical Approach:** Check the squared euclidean distance d_i^2 of each point $\boldsymbol{p}_i = [x_i, y_i, z_i]^T$ in the cloud to the local origin in a greedy fashion and remove it if it is closer than r_{fil}^2.
2) **Kd-tree Approach:** Build a kd-tree from the point cloud and then perform a fixed radius search using r_{fil} around the origin $\boldsymbol{o} = [0, 0, 0]^T$.

We down-sample the points as a part of the filter pre-processing step with a leaf size of d_{ds}, which we determine empirically.

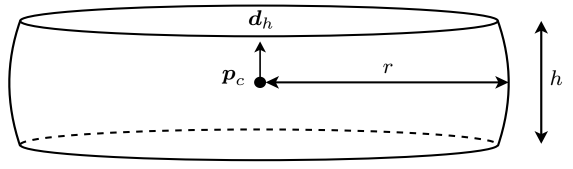


Fig. 4: Height limited sphere. Defined by the center point $\boldsymbol{p}_c$, the vector $\boldsymbol{d}_h$, which defines the orientation, the radius r of the sphere and the height h in $\boldsymbol{d}_h = \boldsymbol{n}$ direction.

2) Creating the Sub-Cloud: Now we want to further exclude potential non-ground points. The points, which represent the ground, are located within a region around the point $\boldsymbol{p}_q$, where the spherical mobile mapping system touches the ground. Let $\boldsymbol{n}$ be the local normal vector of the ground plane from the previous iteration and r_s be the radius of the sphere, then we calculate $\boldsymbol{p}_q$ as:

$$\boldsymbol{p}_q = r_s \, \boldsymbol{n} \, . \tag{1}$$

To determine which points now are the most likely to represent the ground according to $\boldsymbol{p}_q$, there are the following two potential options:

1) **Geometrical Approach:** Consider the shape shown in Figure 4 with radius r and height h. Greedily check each remaining point $\boldsymbol{p}_i$ and keep it if the following two criteria are met:

$$||\boldsymbol{p}_i - \boldsymbol{p}_q||^2 \leq r^2 \, , \quad \text{and} \quad |\left(\boldsymbol{p}_i - \boldsymbol{p}_q\right) \cdot \boldsymbol{n}| \leq \frac{h}{2} \, . \tag{2}$$

2) **Kd-tree Approach:** Build a kd-tree from the filtered point cloud and then find the k-nearest-neighbors to $\boldsymbol{p}_q$.

In this work we compare a total of six different pre-processing combinations (filtering: none, geometrical, or kd-tree; sub-cloud: geometrical or kd-tree).

C. Normal Vector Calculation

While the two plane fitting algorithms (LSF and PCA) directly provide the normal vector of the fitted plane, the plane detection algorithms (RANSAC and RHT) do not. Instead, RANSAC returns the inlier points of the estimated plane. To extract the normal vector, we apply PCA instead of LSF to the inlier points based on [17]. The RHT returns both the plane parameters (distance from the origin ρ, and two angles θ and φ) and the inlier points belonging to that plane. We extract the normal vector $\boldsymbol{n} = [n_x, n_y, n_z]^T$ directly from the plane parameters as follows:

$$n_x = \cos\left(\theta\right) \sin\left(\varphi\right) , \tag{3}$$

$$n_y = \sin\left(\theta\right) \sin\left(\varphi\right) , \tag{4}$$

$$n_z = \cos\left(\varphi\right) . \tag{5}$$

D. Library Selection and Implementation

We use the open-source pcl, which does not implement RHT. Thus, we implement the algorithm ourselves using a ball accumulator based on 3DTK [24], [36]. Furthermore, we use the FLANN implementation of the kd-tree available from pcl. Table I lists all the hyperparameters and their values used in our implementation. After converting the normal vector of the detected plane to the global frame we determine its inclination angle. As an additional check, we assume that the inclination of the ground plane does not exceed a specified maximum angle α_{max}. For plane fitting, we only make a single attempt to detect the ground plane, since there is no reliable way of recovering from a falsely identified ground plane. In contrast, for the plane detection algorithms we perform `max_iterations` attempts, since we are able to simply delete the inlier points from the falsely identified plane and retry. If the algorithms are not able to find a plane, the corresponding frame is discarded and labeled as a fail.

TABLE I: Overview of general parameters for pre-processing and plane segmentation used in the implementation.

Name	Value	Name	Value
Pre-Processing		**RHT**	
Filter radius r_{fil} [m]	0.35	`rhoNum`	7
Leaf size d_{ds} [m]	0.10	`phiNum`	90
Radius sphere r_s [m]	0.145	`thetaNum`	180
Radius sub-cloud r [m]	2.0	`rhoMax`	5
Height sub-cloud h [m]	0.2	`accumMax`	10
k (for k-NNS)	150	`minDist`	0
Plane Segmentation General		`maxDist`	1.8×10^{308}
`max_iterations`	3	Max. iterations	10,000
Incl. angle α_{max} [°]	45	Inlier threshold [m]	0.05
RANSAC			
`dist_thresh` [m]	0.01	`probability_`	0.99

V. EXPERIMENTS

We perform eight experiments across different scenarios to evaluate the different combinations of approaches for pre-processing and normal vector extraction. All datasets have been created on the Computer Science (CS) Campus of the Julius-Maximilians University Würzburg. We design each experiment to highlight specific key aspects such as limited area, varying inclinations, and different movement speeds.

A) **Hallway of the CS building:** The ground plane is as dominant as the walls in the LiDAR data. Leveled, flat floor and slow speed.

B) **Hall of the CS building:** The ground plane is the dominant plane in the LiDAR data. Leveled, flat floor and slow speed.

C) **Wall:** The dominant plane in the LiDAR data is the wall instead of the ground. Leveled, flat floor and slow speed.

D) **Physics building ramp - slow motion:** Environment with two different flat surfaces: the base floor and a narrow ramp with dominant walls. Slow speed.

E) **Physics building ramp - fast motion:** Same scenario as in experiment D, but fast speed.

F) **Outside hill moving up:** No walls, continuously changing inclinations (not flat) and slow speed.

G) **Outside hill moving down:** Same as experiment F, but moving down the hill.

TABLE II: Results of the pre-processing analysis. Average runtimes and time complexity of pre-processing methods, followed by total execution time, averaged across all experiments and plane identification methods.

	Pre-Proc. [ms]	Complexity	Total [ms]
none-geo	**3.862**	$\mathcal{O}(n)$	21.343
none-kdt	13.627	$\mathcal{O}(n \log n)$	15.193
kdt-geo	7.917	$\mathcal{O}(n \log n)$	9.572
geo-geo	6.157	$\mathcal{O}(n)$	**7.959**
geo-kdt	7.720	$\mathcal{O}(n \log n)$	10.031
kdt-kdt	8.951	$\mathcal{O}(n \log n)$	11.122

H) **Tunnel:** No planar ground at all, but a narrow, cylindrical shaped tunnel.

In this work, we refer to slow speed as approximately $2\,\mathrm{rad/s}$, while fast speed corresponds to approximately $5\,\mathrm{rad/s}$. Figure 5 shows the different environments.

VI. EVALUATION

In this section, we evaluate the performance of the different approaches using the following metrics: runtime, accuracy, and robustness. We consider the real-time criterion to be met if the total runtime is faster than $50\,\mathrm{ms}$, which corresponds to the frequency of the LiDAR at $20\,\mathrm{Hz}$. In order to assess the accuracy, we must measure either the ground truth normal vectors of the environment, or the inclination of the slope. We do this by scanning the previously mentioned environments with a RIEGL VZ-400 TLS, which has an angular resolution of $0.0005°$ and a range accuracy of $\pm 5\,\mathrm{mm}$ at $100\,\mathrm{m}$. Using these highly precise maps, we calculate the ground truth normal vectors for each point in the cloud considering its 20 nearest neighbors. As the accuracy measure, we consider the Mean Angle Error (MAE) and Median Angle Error (MedAE) of the inclination between the measured and ground truth normal vector. We calculate the inclination angle α as:

$$\alpha = \cos^{-1}(\boldsymbol{n} \cdot \boldsymbol{g}) \,, \tag{6}$$

where $\boldsymbol{g} = [0,\, 0,\, -1]^T$ is the global normalized gravity vector. As a robustness measure, we use the fail rate of the algorithms, i.e., the percentage of how many frames have been discarded. For the evaluation, we abbreviate the pre-processing combinations as "filtering - sub-cloud" (e.g., none-kdt refers to no filtering with the kd-tree sub-cloud approach).

A. *Pre-Processing Runtime Evaluation*

The pre-processing phase plays an important role in the overall performance of the different approaches in terms of runtime. Table II lists the average runtime of all pre-processing methods, the time complexity notation for each method, as well as the average total runtime across all experiments and plane segmentation algorithms. The results indicate that the none-geo method is the fastest, regarding only pre-processing, with an average runtime of 3.862 ms, whereas the none-kdt method is the slowest with 13.627 ms,

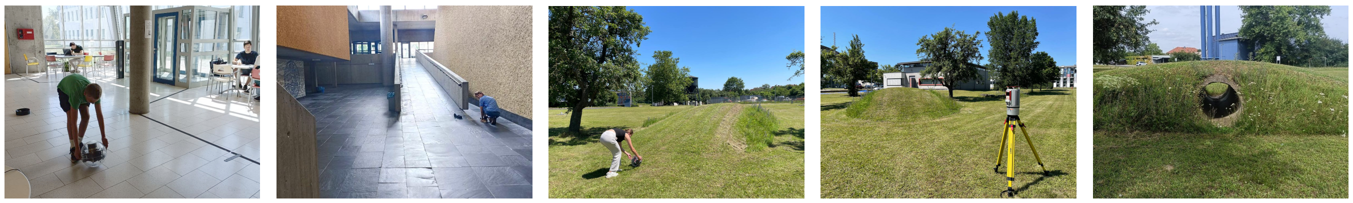
TABLE III: Final results of the plane segmentation analysis. Total average runtime [ms], MAE [deg], MedAE [deg] for the different plane segmentation algorithms under different pre-processing combinations across all experiments. Red: Worst value for plane segmentation algorithm. Green: Best overall combination. Bold: Highest accuracy.

	Avg. Results	LSF	PCA	RAN	RHT
none-geo	**Runtime [ms]**	9.629	26.205	23.212	4.647
	MAE [deg]	13.933	14.616	13.353	12.786
	MedAE [deg]	12.036	13.227	11.769	10.796
	Fail rate [%]	3.795	4.159	2.277	1.967
none-kdt	**Runtime [ms]**	0.244	1.107	1.161	3.138
	MAE [deg]	18.340	17.950	19.413	17.308
	MedAE [deg]	16.137	15.112	17.869	15.267
	Fail rate [%]	42.790	40.896	31.458	46.135
kdt-geo	**Runtime [ms]**	0.378	1.698	1.847	2.468
	MAE [deg]	12.236	12.294	**9.612**	10.516
	MedAE [deg]	10.356	10.392	**8.475**	8.844
	Fail rate [%]	5.951	5.926	3.858	4.481
geo-geo	**Runtime [ms]**	0.415	1.904	2.035	2.600
	MAE [deg]	12.191	12.187	**9.525**	10.432
	MedAE [deg]	10.317	10.228	**8.305**	8.727
	Fail rate [%]	5.940	5.923	3.898	4.320
geo-kdt	**Runtime [ms]**	0.359	1.635	2.781	4.369
	MAE [deg]	13.580	13.584	10.023	**9.560**
	MedAE [deg]	11.170	11.224	8.577	**8.099**
	Fail rate [%]	33.028	33.173	6.047	8.291
kdt-kdt	**Runtime [ms]**	0.334	1.505	2.598	4.153
	MAE [deg]	13.584	13.575	10.049	**9.559**
	MedAE [deg]	11.267	11.320	8.604	**8.159**
	Fail rate [%]	33.076	33.211	6.063	8.253

which is due to the additional overhead added by the construction of the kd-tree with $30,000$ points. Although the none-geo method is the fastest regarding the pre-processing runtime, it is the slowest in terms of total runtime with 21.3 ms, which is due to the missing down-sampling of the points for the plane identification phase. Thus, the geo-geo pre-processing leads to the fastest overall runtime of 7.959 ms.

B. *Plane Identification Analysis*

Table III offers a comprehensive summary of the plane identification analysis, showing the total average runtime [ms], MAE [deg], MedAE [deg], and fail rate [%] for each plane segmentation algorithm under different pre-processing combinations across all experiments. The none-kdt method consistently shows the highest fail rates ($> 31.4\,\%$), as well as MAEs and MedAEs ($> 13.5\,°$), because of the increased non-ground point density. Using geo-geo pre-processing followed by RANSAC to detect planes offers the best compromise across runtime, fail rate, and accuracy. The results suggest that the geometrical sub-cloud creation methods result in lower overall runtime for all plane detection algorithms (RANSAC and RHT), due to the better representation of the ground plane. In addition, we observe significantly lower fail rates for the geometrical sub-cloud creation compared to the kd-tree method. The sub-cloud approach affects LSF and PCA even stronger, where the kd-tree approach has a fail rate of $33.1\,\%$ compared to $5.9\,\%$ for the geometrical approach, and further leads to increased MAEs and MedAEs by $10\,\%$ and $7\,\%$, respectively. This is expected due to the higher outlier-to-ground ratio in the kd-tree sub-cloud. In



(a) Experiment A, B, & C: The CS building. (b) Experiment D & E: Physics ramp. (c) Experiment F: Outside hill moving up. (d) Experiment G: Outside hill moving down. (e) Experiment H: Tunnel.

Fig. 5: Environments used for the experiments, featuring inside and outside environments with varying ground inclinations.

summary, the geometric sub-cloud approach outperforms the kd-tree method due to the significantly higher robustness and the lower total runtime. Regarding the accuracy, we see higher MAEs and MedAEs ($> 10.0\,°$) for the plane fitting algorithms compared to the plane detection algorithms. This is also an expected result, since it is likely that not all non-ground points have been removed in the pre-processing steps. Consequently, we determine that relying solely on plane fitting algorithms is not a suitable solution for our application. Conclusively, RANSAC is the most accurate for geometrical sub-clouds, while RHT works almost as good with kd-tree sub-clouds, showing an insignificant difference in MedAE of $0.3\,°$. Figure 6 shows the absolute angle errors of all individual experiments as a time series, using geo-geo pre-processing followed by RANSAC plane detection. Interestingly, the errors are of oscillating nature and are more coherent with the rolling motion of the sphere, than with environmental factors. Whenever the LiDAR is in a vertical position, that is, the ground is barely visible containing only a few points, the errors are greater.

VII. CONCLUSIONS

We have thoroughly addressed the real-time on-board calculation of the ground normal from 3D-LiDAR-inertial data on our spherical mobile mapping system. Based on the findings from the previous section, we recommend to use geometrical filtering and sub-cloud creation as pre-processing steps, followed by RANSAC to identify the inlier points of the ground plane and fitting a plane through those points with PCA. We have proven real-time capability and robustness, with average angle errors of 9°. Needless to say, a lot of work remains to be done. We want to decrease the average error further by considering not only individual LiDAR frames, but build an online globally consistent map, either by using a sliding window across past scans, or by utilizing key frames. Furthermore, we need to include the resulting normal vector in our odometry pipeline and test the implications on that subsystem. Additionally, especially considering more complex non-planar surfaces, we want to extend our RANSAC algorithm with more models, e.g., to account for cylindrical tunnels, pipes, or ramps. Moreover, future work includes the development of an online certainty measure to assess the reliability of the normal vector calculation beyond just failure or success. Nevertheless, this research pushes the under-explored field of spherical mobile mapping systems further towards autonomy.

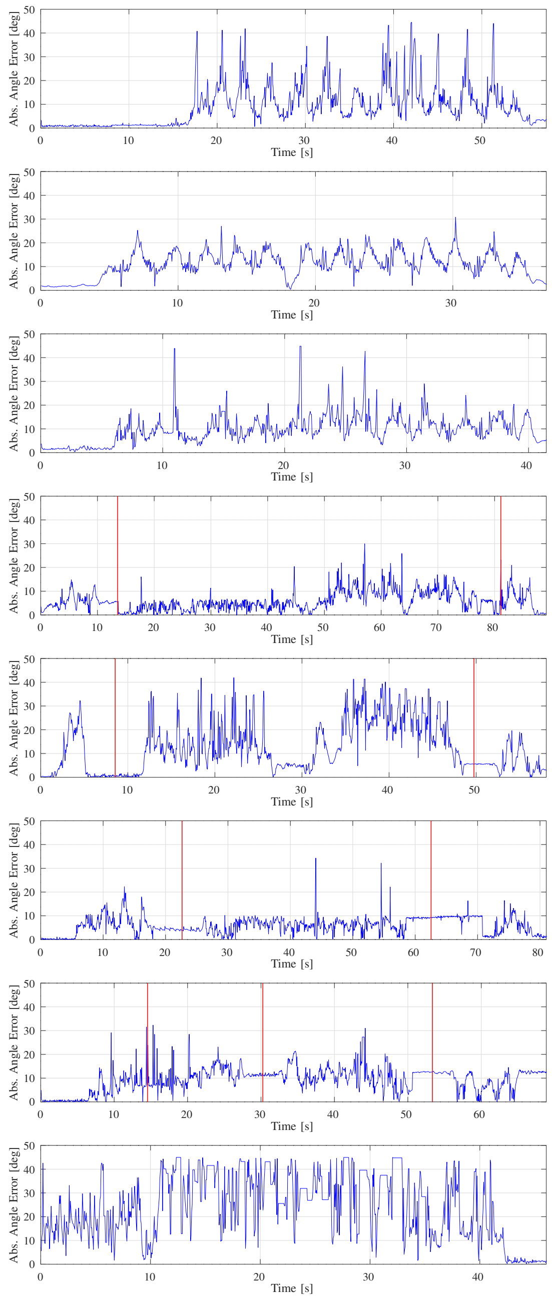


Fig. 6: Top to bottom: Experiments A-H. Absolute angle error for RANSAC using geo-geo pre-processing. Red lines indicate changing ground inclination.

References

[1] C. Bösch and F. Arzberger, "GitHub: Spherical Ground Finder," https://github.com/fallow24/Spherical-Ground-Finder.git, 2024, accessed: 15.09.2024.

[2] A. Nüchter, *3D robotic mapping: the simultaneous localization and mapping problem with six degrees of freedom*. Springer, 2008, vol. 52.

[3] J. L. Vermeulen, A. Hillebrand, and R. Geraerts, "A comparative study of k-nearest neighbour techniques in crowd simulation," *Computer Animation and Virtual Worlds*, vol. 28, no. 3-4, p. e1775, 2017.

[4] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nuechter, "Comparison on nearest-neigbour-search strategies and implementations for efficient shape registration," *Journal of Software Engineering for Robotics (JOSER)*, vol. 3, pp. 2–12, 01 2012.

[5] A. Nguyen and B. Le, "3d point cloud segmentation: A survey," in *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*. IEEE, 2013, pp. 225–230.

[6] C. M. Huang and Y.-H. Tseng, "Plane fitting methods of lidar point cloud," in *29th Asian Conference on Remote Sensing 2008, ACRS 2008*, 2008, pp. 1925–1930.

[7] A. Nurunnabi, D. Belton, and G. West, "Robust statistical approaches for local planar surface fitting in 3d laser scanning data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 96, pp. 106–122, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271614001762

[8] R. Hoffman and A. K. Jain, "Segmentation and classification of range images," *IEEE transactions on pattern analysis and machine intelligence*, pp. 608–620, 1987.

[9] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: https://doi.org/10.1080/14786440109462720

[10] A. Nurunnabi, D. Belton, and G. West, "Diagnostics based principal component analysis for robust plane fitting in laser data," in *16th Int'l Conf. Computer and Information Technology*. IEEE, 2014, pp. 484–489.

[11] N. Vaswani and P. Narayanamurthy, "Static and dynamic robust pca and matrix completion: A review," *Proceedings of the IEEE*, vol. 106, no. 8, pp. 1359–1379, 2018.

[12] M. Hubert, P. J. Rousseeuw, and K. Vanden Branden, "Robpca: a new approach to robust principal component analysis," *Technometrics*, vol. 47, no. 1, pp. 64–79, 2005.

[13] P. Netrapalli, N. UN, S. Sanghavi, A. Anandkumar, and P. Jain, "Non-convex robust pca," *Advances in neural information processing systems*, vol. 27, 2014.

[14] Y. Chen and E. Candes, "Solving random quadratic systems of equations is nearly as easy as solving linear systems," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[15] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust pca via gradient descent," *Advances in neural information processing systems*, vol. 29, 2016.

[16] H. Xu, C. Caramanis, and S. Sanghavi, "Robust pca via outlier pursuit," *Advances in neural information processing systems*, vol. 23, 2010.

[17] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *2009 IEEE international conference on robotics and automation*. Ieee, 2009, pp. 3206–3211.

[18] W. Song, L. Zhang, Y. Tian, S. Fong, and S. Sun, "3d hough transform algorithm for ground surface extraction from lidar point clouds," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2019, pp. 916–921.

[19] P. V. Hough, "Method and means for recognizing complex patterns," Dec. 18 1962, uS Patent 3,069,654.

[20] G. Vosselman, "Building reconstruction using planar faces in very high density height data," *International Archives of Photogrammetry and Remote Sensing*, vol. 32, no. 3; SECT 2W5, pp. 87–94, 1999.

[21] R. C. Bolles and M. A. Fischler, "A ransac-based approach to model fitting and its application to finding cylinders in range data." in *IJCAI*, vol. 1981, 1981, pp. 637–643.

[22] F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer, "Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data," in *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, vol. 36, 2007, pp. 407–412.

[23] R. Hulik, M. Spanel, P. Smrz, and Z. Materna, "Continuous plane detection in point-cloud data based on 3d hough transform," *Journal of visual communication and image representation*, vol. 25, no. 1, pp. 86–97, 2014.

[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, pp. 1–13, 2011.

[25] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[26] F. Yang, M. Jiang, and C. Xu, "Real-time ground-plane refined lidar slam," *arXiv preprint arXiv:2110.11517*, 2021.

[27] M. Guo, L. Zhang, X. Liu, Z. Du, J. Song, M. Liu, X. Wu, and X. Huo, "3d lidar slam based on ground segmentation and scan context loop detection," in *2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2021, pp. 692–697.

[28] X. Wei, J. Lv, J. Sun, and S. Pu, "Ground-slam: Ground constrained lidar slam for structured multi-floor environments," *arXiv preprint arXiv:2103.03713*, 2021.

[29] W. Wen and L.-T. Hsu, "Agpc-slam: Absolute ground plane constrained 3d lidar slam," *NAVIGATION: Journal of the Institute of Navigation*, vol. 69, no. 3, 2022. [Online]. Available: https://navi.ion.org/content/69/3/navi.527

[30] D. Liu, H. Ni, X. Zhou, N. Yang, and W. Yan, "Flexible ground constrained lidar slam with a novel plane detection," *Computers and Electrical Engineering*, vol. 117, p. 109287, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790624002155

[31] Hesai Technology Co., Ltd., "PandarXT-32: 32-Channel Medium-Range Mechanical LiDAR User Manual," http://www.oxts.com/wp-content/uploads/2021/01/Hesai-PandarXT_User_Manual.pdf,, 2020, accessed: 08.08.2024.

[32] BMAX, "Maxmini B3 Plus," https://www.bmaxit.com/Maxmini-B3-Plus-pd722218588.html,, 2024, accessed: 08.08.2024.

[33] J. Zevering, A. Bredenbeck, F. Arzberger, D. Borrmann, and A. Nuechter, "Imu-based pose-estimation for spherical robots with limited resources," in *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2021, pp. 1–8.

[34] P. Heinisch, F. Arzberger, and A. Nüchter, "Inertial lidar motion distortion correction on spherical mobile mapping systems for planetary exploration," in *Proceedings of the International Conference on Space Robotics (iSpaRo '24)*, Luxembourg, May 2024. [Online]. Available: https://robotik.informatik.uni-wuerzburg.de/telematics/download/iSpaRo2024.pdf

[35] E. Maltezos and C. Ioannidis, "Automatic extraction of building roof planes from airborne lidar data applying an extended 3d randomized hough transform," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3, pp. 209–216, 2016.

[36] A. Nüchter and K. Lingemann, "3DTK—The 3D Toolkit. 2011," https://slam6d.sourceforge.io/index.html, 2011.