remote sensing

MDPI

Article

Localization in Unstructured Environments: Towards Autonomous Robots in Forests with Delaunay Triangulation

Qingqing Li *, Paavo Nevalainen, Jorge Peña Queralta, Jukka Heikkonen and Tomi Westerlund

Turku Intelligent Embedded and Robotic Systems, University of Turku, 20500 Turku, Finland; ptneva@utu.fi (P.N.); jopequ@utu.fi (J.P.Q.); jukhei@utu.fi (J.H.); tovewe@utu.fi (T.W.)

* Correspondence: qingqli@utu.fi

Received: 7 May 2020; Accepted: 5 June 2020; Published: 9 June 2020

Abstract: Autonomous harvesting and transportation is a long-term goal of the forest industry. One of the main challenges is the accurate localization of both vehicles and trees in a forest. Forests are unstructured environments where it is difficult to find a group of significant landmarks for current fast feature-based place recognition algorithms. This paper proposes a novel approach where local point clouds are matched to a global tree map using the Delaunay triangularization as the representation format. Instead of point cloud based matching methods, we utilize a topology-based method. First, tree trunk positions are registered at a prior run done by a forest harvester. Second, the resulting map is Delaunay triangularized. Third, a local submap of the autonomous robot is registered, triangularized and matched using triangular similarity maximization to estimate the position of the robot. We test our method on a dataset accumulated from a forestry site at Lieksa, Finland. A total length of 200 m of harvester path was recorded by an industrial harvester with a 3D laser scanner and a geolocation unit fixed to the frame. Our experiments show a 12 cm s.t.d. in the location accuracy and with real-time data processing for speeds not exceeding 0.5 m/s. The accuracy and speed limit are realistic during forest operations.

Keywords: robotics; localization; delaunay triangulation; SLAM; forest localization

1. Introduction

Over the last ten years, autonomous harvesting and transportation have become the long-term goal of the future development of the forest industry and attracted the interest of the research community. Naturally, there are several intermediate goals such as the organization of the public data [1], and the gradual increase of autonomy in varying degrees, starting with short-range transport in forests [2]. Incremental advances in forest autonomy include driver assistance platforms and function-specific automation. For example, these include the automated selection of tree stems to be processed, micro-tasks such as sequencing the processing of individual trees, local route planning, or semi-autonomous transportation and quality assurance. In all these tasks, one key element is the availability of tree maps, together with methods enabling the identification and selection of individual trees. In addition, local route planning requires accurate updates on the position of the harvester inside the forest in real-time, which can not be relied on global navigation satellite system (GNSS) sensors [3,4].

This work focuses on the real-time localization of an autonomous forwarder unit, a forestry vehicle that collects felled logs and hauls them to a local loading area. The required transport distance is usually short, in the range of 100–400 m, and happens along a rough-terrain track a forest harvester has previously made while performing logging. To autonomously navigate in the forest, visual sensors present significant challenges owing to the lack of a stable background from which contours could be extracted, as well as demanding low-light and harsh weather conditions [5]. Forest environments present difficult light conditions during winter and at night, therefore we considered only the lidar technology. Nonetheless, it is worth noting that visual-inertial odometry [6], and other methods for estimating structure from motion [7], have seen significant advances over the past few years. For instance, in [7] the authors report improvements in day-night recognition stability. This work required long range up to 60 m for location and wide field of view, and we have not tested any approach based on visual sensors.

There are multiple well-established frameworks and algorithms for autonomous driving in urban environments [8], as well as localization and mapping in roads and buildings [9]. Many of the solutions proposed in these areas have a strong dependency on lidar scanners [10], among other sensors. In the field of forest mapping and navigation, several researchers have utilized terrestrial laser scan (TLS) to build point-cloud maps [11–15]. In some of these works [11,13], data are collected from fixed-positioned tripods to gain an understanding of how well individual trees can be detected and their diameter at the breast height (DBH) can be registered. Other studies rely on mobile laser scanning (MLS) [12,14,15] to simulate the operation of the harvester.

A forwarder is a forestry vehicle collecting the felled logs to a storage positions. An autonomous path tracking based on GPS only was demonstrated in [16] using an industrial vehicle in a scene with clear-cut forest. To actually operate autonomously, and to locate and collect logs, more complex environments need to be analyzed. A summary of autonomous robots in cross-country terrain is [17], which includes also military applications. There seems to be not many attempts to have autonomous forwarders under full canopy.

Navigating in a forest presents several inherent challenges owing to the complexity and lack of structure in the environment [18]. A realistic forest harvesting process involves constant obstructions from cut or fell branches and trunks and irregular microrelief and terrain. Furthermore, the movement of a harvester tends to be rotational most of the time, with constant changes in orientation along the work cycle and very slow translational motion or even with no movement at all while trees are being cut. These particularities of forest navigation bring both advantages and disadvantages to standard development of autonomous mobile robots: idle periods and slow motion aid at realizing real-time operation and data processing, while fast rotations hinder accurate matching of scan data (odometry) owing to a large variability in the distance between near and far objects, and multiple similar objects in different directions. A small error in the orientation estimation can significantly affect the mapping of trees that are farther away [12,19].

1.1. Research Hypotheses and Objectives

Research in the field of localization and mapping for autonomous robots can be roughly divided into two approaches: simultaneous localization and mapping (SLAM), and sequential location and mapping. The main difference between these two is that SLAM has the location and mapping occurring at the same time, whereas the sequential version has the mapping part completed after the track of the vehicle has been constructed for a considerable length. Localization is often done by finding either a partial match between sensor data batches acquired at different times, or finding a spatial transformation that maximizes the data matching. A different approach is then to classify localization methods based on whether the matching process is done globally or locally. The same can be applied to different types of sensor data, including images [20]. Nonetheless, we are interested in matching point clouds generated from lidar scanners.

Locally converging methods have a valley of convergence [21], which is sometimes defined e.g., by step length limit and the point cloud overlap limit, therefore we study a global point cloud matching approach.

In this paper, our objective is to enable fast, real-time global localization for autonomous forest harvesters and forwarders. Our experiments have been done using data from a 3D lidar scanner situated in the front panel of a manually operated forwarder. The methods presented in this paper rely on the assumption that a global forest map in the form of a point cloud is available. In practice, we first create the map using state-of-the-art lidar odometry and mapping algorithms, and then evaluate our localization approach. The localization is global and does not depend on previous states or position estimations, therefore providing a robust solution for long-term autonomy.

1.2. Localization in Autonomous Mobile Robots

Autonomous mobile robots meant to operate outdoors often rely on GNSS sensors as the basic source of global localization data, and then integrate other sensor data through sensor fusion techniques for local position estimation [10]. GNSS sensors alone do not provide enough accuracy in urban or dense environments, with accuracy often in the order of meters [22]. While GNSS sensors have increased their accuracy in recent years, multiple challenges remain in environments with structures affecting the signal path. In particular, GNSS signals are weak in forests, owing to the irregular land contours and the coverage that tree foliage provides [23]. Therefore, we focus on the design and development of localization methods that rely solely on lidar data, with GNSS providing only the initial position before entering the forest or starting the autonomous operation. The following are the main approaches enabling the estimation of the movement of a mobile robot based on lidar scans by either comparing consecutive scans (hereinafter also called point clouds) or a given scan with a global point cloud. We refer to comparing scans as scan matching:

Full scan matching. In SLAM algorithms, a necessary step previous to update the map is to estimate the relative movement of the robot with respect to its last known position. This is equivalent to finding a geometrical transformation between the corresponding point clouds. One of the most widely utilized methods for calculating such transformation is the iterative closes point (ICP) algorithm [24,25]. In ICP, pairs of points between the two point clouds being compared are iteratively selected until a transformation with acceptable pairing error distance is found. In this case the acceptable error is chosen to be 0.6 m, which is 30% of the mean nearest neighbor tree distance. The mean nearest neighbor distance is a dynamical entity (2.0 m in this case), which can be observed from the global map in each case.

Several variants of the ICP algorithm are available through the open-source point cloud library (PLC) [26]. Other popular methods include the perfect match (PM) [27], or normal distribution transforms (NDT) [28] algorithms. Most of these algorithms can be extended and utilized for map-based localization, where a global point cloud is available and a given lidar scan is matched with only a subset of the global map. However, in these cases, an estimation of the previous position is needed, since otherwise a standard ICP routine could enter a global search phase, which is computationally much more costly due to convergence safeguards, etc. See e.g., [29] about the intricacies of the guaranteed global convergence. Since our objective is to enable global localization without complete dependence on previous states, we develop a novel method that takes into account the geometry of the environment instead of using the full point cloud.

Feature-based matching. Generic point cloud matching algorithms do not take into account the structure of any potential features within the scans that are being compared. When the environment where robots operate has known structures, feature-based scan matching can significantly enhance the accuracy of the matching process [30–32]. Moreover, owing to the preprocessing step in which raw data is transformed into features, the computational time required to calculate the transformation can be reduced. In this direction, Zhang et al. presented the lidar odometry and mapping (LOAM) method [33], which assumes a structured environment where planes and edges can be detected. Then, the transformation between two point clouds can be estimated by aligning the planar and edge feature points from each of them. Multiple algorithms have extended the original LOAM method for other types of features. Among them, Shan et al. presented the ground-optimized Lego-Loam [34], which delivers optimized real-time three-dimensional point cloud matching in small scale embedded devices. While feature-based matching algorithms yield accurate results, their applicability is limited in forests, with most methods focusing on urban or indoor environments. In forests or other unstructured environments, edge and planar features are either noisy or missing. In this paper, we still rely on the state-of-the-art Lego-Loam algorithm for building a map of the forest prior to the operation of the robots. This process is done offline. However, we develop an alternative method for localization because of our needs for real-time operation and global localization.

Geometrical matching methods. Geometry-based localization algorithms have the potential to recognize the position in constant time, which is an attractive property towards real-time localization with embedded processors. Geometrical methods are often applied to graph-like structures to find geometric transformations that match either the complete graph or a set of subgraphs. Thrun et al. developed the notions of sparse extended information filters (SEIFs) which exploit the structure inherent through the local web-like networks of features maps [35,36]. In a similar direction, Ulrich et al. proposed a global topological representation of places with object graphs serving as local maps for place classification and place recognition [37–39]. Among the most relevant approaches to our work are place-recognition algorithms, such as Bosse's work on a keypoints similarities voting method in 2D and 3D lidar data [40–42].

1.3. Contribution and Structure

Most of the aforementioned point cloud matching algorithms focus on structured urban environments, or trajectory tracking through fixed, predefined paths such as roads. Nonetheless, their performance is significantly degraded in unstructured environments and, in particular, in forests. In a forest harvesting operation, harvesters and forwarders travel through undefined paths. Moreover, the detection of potential features (ground and tree stems) is considerately affected by irregularities (branches and foliage in the trees, and small plants, rocks or fell trees in the ground). While some of the state-of-the-art methods yield reasonably good results for mapping and odometry [12,34,43], our aim is to develop a real-time method for global localization based on known geometrical features of forest environments. Therefore, in this work, we pursue the design and development of a localization method which is reliable, accurate and of low computational cost. The computation is aimed to occur in real time on-board of harvesters and forwarders in the forest with canopy cover. The final hardware to be utilized in a real application would have similar capabilities to the hardware we have used in the experiments reported in this paper (see the Methodology chapter for more details).

Relatively randomly located but ubiquitous tree stems are a natural basis for localization efforts. We also assume that a map of the forest in the form of a point cloud is available before the mission starts.

We propose a lightweight and geometry-based localization method for pose estimation within a global map of a harvester in dense and unstructured forest environments. The proposed method is lightweight and matches a triangularization of a single lidar scan with a subgraph of the triangularization of the global forest map. Compared to matching raw lidar data point-by-point, we reduce the problem to a triangularization created based on the positions of tree stems. This allows us to reduce the amount of point to be matched to approximately 1% of the size of a single lidar scan because individual trees get an average of 100 laser hits in our setting: a 16-channel 3D lidar with a field of view of about 210° and situated at a height of 1.5 m. Closer trees get, however, considerably higher amounts of laser hits. The proposed method is based on triangle dissimilarity minimization, and that is why we say the method is geometry-based. To the best of our knowledge, this is the first paper to utilize Delaunay triangulation for the purpose of localizing a vehicle in a forest environment. Moreover, we provide a fully experimental approach with a realistic use case in forest harvesting. The main contributions of our work are the following:

- the introduction of a Delaunay triangulation graph map for localization in forests; and
- the design and development of a framework to solve the vehicle tracking problem in a local coordinate system relying solely on lidar data, without GNSS or inertial sensor data.

The remainder of this paper is organized as follows. Section 2 introduces the study area and equipment, the optimization problem to be solved and the methodology we have followed. Section 3 reports experimental results. In Section 4, we discuss the potential improvements, as well as the main benefits of our approach. Finally, Section 5 concludes the work and outlines future research directions.

2. Material and Methods

In this section we introduced the proposed localization algorithm and related background concepts.

2.1. Study Area

The study area covers one mature pine stand ready for second thinning, which represents a typical second thinning Finnish forest. This second thinning increased the mean distance of trees from 4.4 m to approximately 8 m judging by a before and after point cloud maps. The location of the study area is illustrated in Figure 1. The terrain profile was mostly flat, the maximum height difference over the site was 8 m on the covered area of approximately 200 m by 300 m. The overall trail length is approximately 200 m covering roughly 2200 trees.

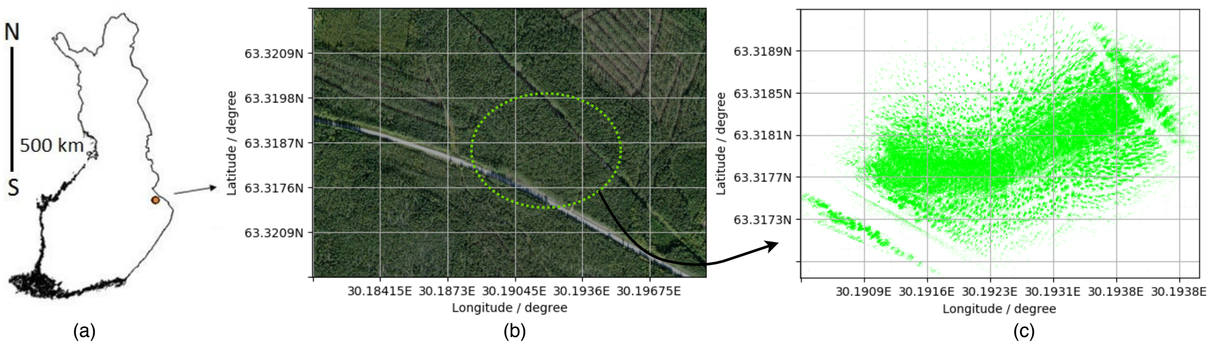

Figure 1. (**a**) The test site in Lieksa, Eastern Finland. (**b**) The forest canopy map from Google maps. (**c**) Point cloud map of the study area.

2.2. System Hardware

The mobile platform was a Komatsu Forest 931.1 forest harvester with a GPS unit and the lidar unit attached to the top of the cabin. The harvester has physical dimensions (length, width, height) of 7.6 m by 2.9 m by 3.9 m, and its mass is 19,610 kg.The maximum driving speed is 8 km/h off-road and 25 km/h on road.

The on-board lidar is Velodyne-16, a 16 channel lidar with 3 cm distance accuracy, 360° horizontal field of view, 30° vertical field of view with ±15° up and down, 5–20 Hz scanning frequency and 100 m scan range. As Figure 2 shows, the lidar unit is fixed on the front of the harvester with a 210° horizontal view. The harvester has a folding frame, which means the point cloud frames scanned have constantly alternating horizontal orientation during the work cycle.

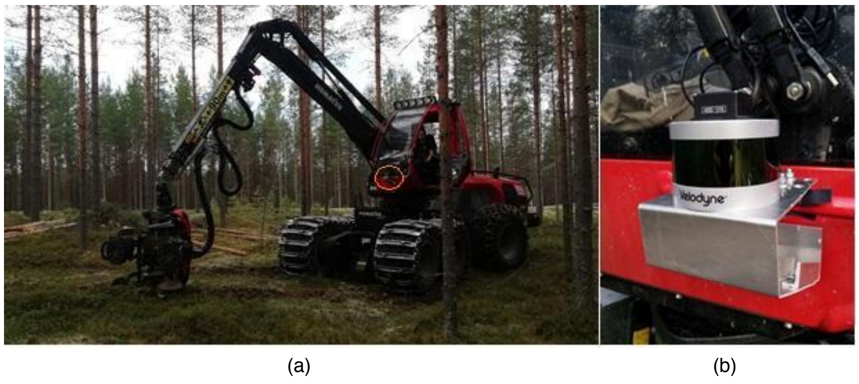

Figure 2. (**a**) Komatsu Forest 931.1 forest harvester. The lidar scanner is marked by a red circle in the front of the windshield of the cabin. (**b**) A close-up of the lidar scanner.

The GPS data was recorded by Spectra SP60 GPS unit. This unit fully utilized all 6 GNSS systems: GPS, GLONASS, BeiDou, Galileo, QZSS and SBAS. In SBAS (WAAS/EGNOS/MSAS/GAGAN) mode, the horizontal position error smaller than 50 cm, and the vertical error is smaller than 85 cm. In differential GPS mode, the accuracy is able to reach 25 cm in horizontal and 50 cm in vertical accuracy. In Real-Time Kinematic position (RTK) mode, the accuracy can reach 8 mm in horizontal and 15 mm in vertical accuracy. Nonetheless, these values are not achievable under dense foliage in a forest environment. Regarding the computing platform, the proposed methods have been tested on an Intel Core i7-9700K CPU (8 cores, up to 3.60 GHz), and 16 GB of RAM. The GPU was not utilized for the implementation of any of the algorithms involved in the localization process introduced in this paper.

This study concerns the development of a localization algorithm for a forwarder unit to use a tree map during autonomous navigation under potentially heavy canopy. For that purpose, the forwarder scans were simulated from the existing lidar produced by a forest harvester. The harvester fells trees and a sector of the view is constantly obstructed by a tree being processed. Effects of this disappear when the initial tree map is being constructed using normal SLAM procedures. This study concerns the utility of the local scans done by a forwarder to be used in comparison to the existing map.

2.3. Methodology

Figure 3 shows an overview of our proposed method. The complete system takes as a sole input the point cloud data from the 3D lidar, and outputs the position and orientation estimation in the reference of the given global map. The system can be divided into six steps as illustrated in Figure 4. The former two steps are done offline by processing the point cloud data defining the global forest map. These steps have a computational complexity that grows linearly with the number of lidar points (first step) and with the number of trees in the map (second step). The latter four steps are then done online on-board the harvester to estimate its position in real time. The steps are:

1. Point-cloud trunk segmentation from the global map (offline).
2. Delaunay triangulation of the global map from the segmented trunk points (offline).
3. Aggregation of 3D lidar scans into a local point cloud defining the robot's position (online).
4. Segmentation of trunks from the local point cloud (online).
5. Delaunay triangulation from the local segmented trunk points (online).

6. Estimation of a geometrical transformation that matches the local Delaunay triangulation with a subset.

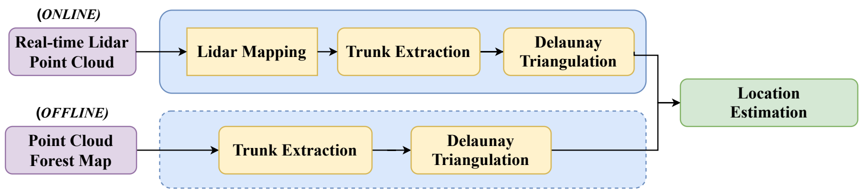

Figure 3. System overview: sensor data is matched with a global map to produce a global position estimation. Consecutive runs are independent.

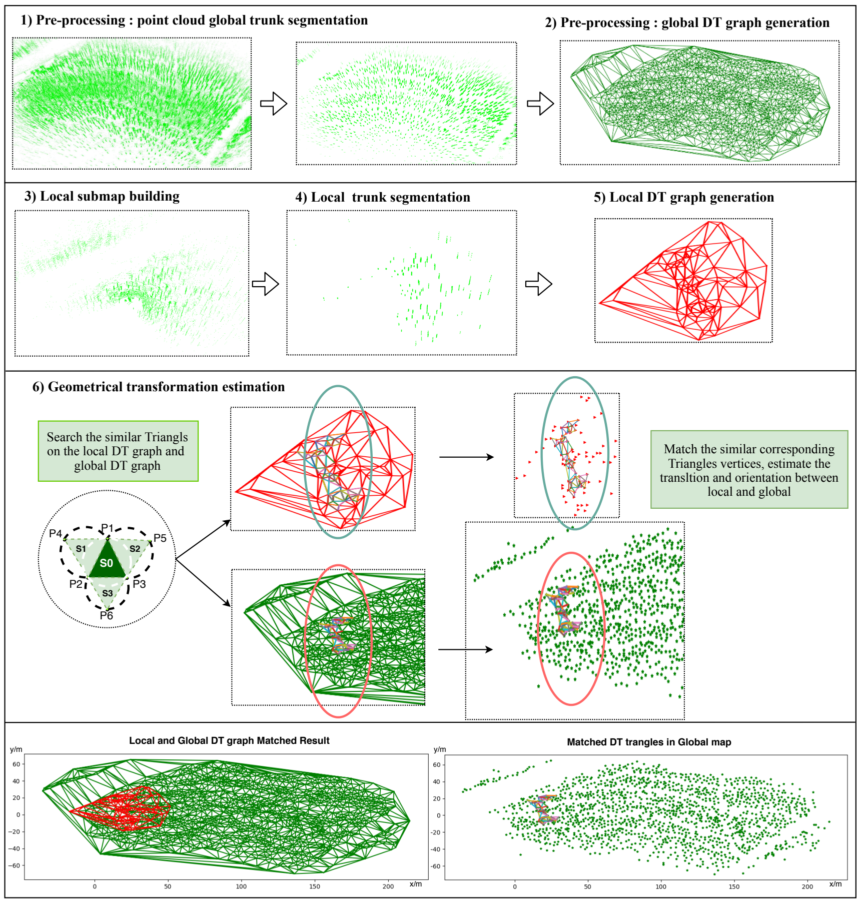
1) Pre-processing : point cloud global trunk segmentation
2) Pre-processing : global DT graph generation
3) Local submap building
4) Local trunk segmentation
5) Local DT graph generation
6) Geometrical transformation estimation
Search the similar Triangls on the local DT graph and global DT graph
Match the similar corresponding Triangles vertices, estimate the transltion and orientation between local and global
P1
P2
P3
P4
P5
P6
S0
S1
S2
S3
Local and Global DT graph Matched Result
Matched DT trangles in Global map
y/m
x/m

Figure 4. Schematic representation of the proposed method to predict the transformation between a local observation sub-map and the global map.

A 2D Delaunay triangulation (DT) process [44] takes in a set of planar points $V \subset \mathbb{R}^2$ and produces a triplet $G = (V, E, T)$, where $E \subset V^2$ is a set of edges and $T \subset V^3$ is a set of triangles with the so called Delaunay property i.e., the circumscribed triangle associated with each triangle $t \in T$ contains no points $v \in V$ others than the three vertex points of the triangle t.

The first step in the proposed method takes a global point cloud $PC_{map} = \{P_i\}_{i=1...n_m} \subset \mathbb{R}^3$, representing the map of the forest, and produces a robust set of trunk positions $PC_{trunks} = \{P_i\}_{i=1...n_t} \subset \mathbb{R}^2$. The accuracy of these positions depends on the accuracy of the lidar being utilized and the size of the trees, as they are calculated as the Euclidean average of all lidar points defining a tree trunk. Note that PC_{trunks} is not a subset of PC_{map}. The second step in our method then subjects the horizontal plane projection PC_{map} to the Delaunay triangularization graph $G_{map} = (V_{map}, E_{map}, T_{map})$ for matching. As we mentioned earlier, both of these steps happen offline before the robot is deployed, or whenever the robot gets a global map update. A global map update only happens when the robot changes to a new location, or enters an area from which it previously had no information.

The third, fourth and fifth steps, which happen online, cover the generation of a local, real-time Delaunay triangulation of the trees within the field of view of the robot. First, we accumulate and aggregate several raw point cloud scans from the 3D lidar and generate a local point cloud $PC_{local} \subset \mathbb{R}^3$. The aggregation relies on real-time lidar odometry from LOAM [33]. Then, following the same procedure as with the global map, we generate a robust two-dimensional set of trunk positions $L_{trunks} \subset \mathbb{R}^2$. From the set of trunks, we can define the local DT graph $G_{local} = (V_{local}, E_{local}, T_{local})$.

Finally, in the sixth step of the process we calculate a rigid body transformation (also called Euclidean transformation or Euclidean isometry), defined by a rotation θ and a translation p_t, between the local DT graph G_{local} and a subset of the global DT graph $G_{match} \subset G_{map}$. The transformation relates directly to the robot position and orientation in the global map. Instead of matching large quantities of the 3D point cloud, the proposed system seeks to match a triangularized representation of the local data against a similar precomputed triangularized global representation. The resulting process is computationally efficient and with a low memory demand.

The approach allows noise in the trunk detection in that the set of triangles that we consider for matching are selected to be best matches (different sets at each location), and the final transformation is averaged by this set.

In the rest of this section, we further describe the process outlined above and delve into the details of the most critical steps. In general, the key idea is building a unique 2D graph topology from the 3D point cloud map, and finding the best match between local and global topology.

2.4. Trunk Point Cloud Segmentation

With a given map PC_{map} of the forest, an essential step in the proposed method is to extract the trunks points that define PC_{trunks} as a set of landmarks from PC_{map}. Compared to the other environmental features such as branch structures, forest floor vegetation and the bushes in the forest, tree trunks are a natural choice as geometric features.

To segment the trunk points, the first step is extracting the trunk point cloud PC_{trunks} from the map point cloud PC_{map}. We employ the Kd-Tree space partitioning structure to accelerate the neighborhood search [45], together with a Euclidean Cluster to find the trunk point cloud's [46]. Instead of focusing on finding every trunk in the PC_{map}, we focus on extracting the most significant ones. We define the most significant trunks in this paper as those that show observational stability, i.e., they can be assumed to be easy to observe by the robot in the near future locations. Thus, we do not consider small trunks or bushes, which are not a worthwhile computational investment to locate and give them a landmark status. As a typical trunk is an approximately vertical object, we assume that the point from a stable trunk has a neighboring point located 2 m above it. This naturally limits the detected trunks, but allows the rest of the algorithm to proceed, since it is enough to get a number of matches from the most prominent objects for the further triangle matches to succeed. Another point is that the trunk detection scheme can be altered in a modular fashion in the future.

As outlined in Algorithm 1, our objective extract all the tree points which are above the ground from PC_{map}. We traverse all points in the PC_{map}, and for each point $p_i = (x_i, y_i, z_i) \in PC_{map}$ we find the nearest neighbor of a hypothetical point $p_i' = p_i + e_3 t_h$, where e_3 represents the vertical unit vector and $t_h = 2\,\mathrm{m}$

above the point p_i. For all such points p_i which do have other points above them, we set the z-axis value to zero and add them to the set P_t. This latter set thus contains the projections of all the points belong to the tree into two dimensions. Following this, we remove the branch points in P_t. To do this, we traverse all points in P_t, and find all the points p that have at least one other neighbor point at a distance less than $t_H < 5\,\text{cm}$. All points meeting the aforementioned condition are added to the set PC_t. Then, we form Euclidean clusters (formed by close points) and drop the clusters that have a size in points smaller than a threshold value $cluster\_threshold$. This value has been defined manually taking into account the existing knowledge on the type of trees in the objective forest area. If larger or smaller trees of interest are present in the forest, then this must be taken into account and the value of $cluster\_threshold$ adjusted accordingly. Finally, we compute the mean $\bar{p} \in \mathbb{R}^2$ of the horizontal projections of the cluster points to represent a landmark trunk. These landmarks then compose the 2D trunk map M_{trunks}. Figure 5 show the proposed trunk point cloud segmentation result. Figure 5a shows the map to be processed, Figure 5b shows the extracted point cloud after removing the ground point cloud and branch point cloud from the input map. Figure 5c shows the final extracted stable trunk point cloud after cluster processing.

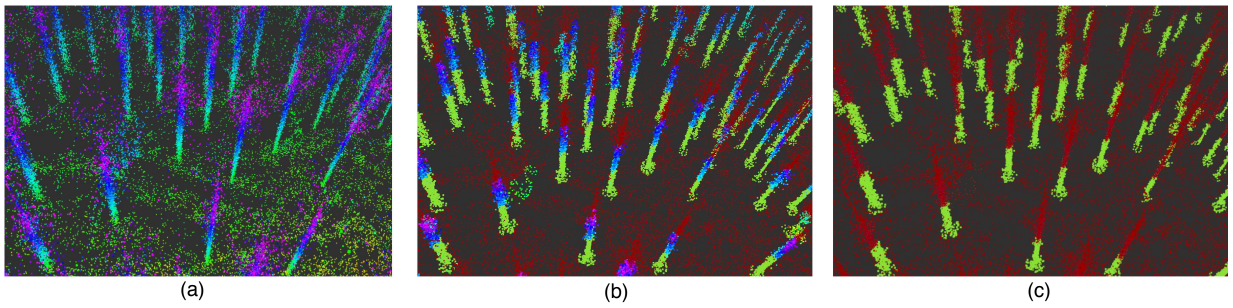

Figure 5. Trunk point cloud segmentation approach. (**a**) The original point cloud map. (**b**) The extracted trunk point cloud (blue and green) after removing ground and branch point cloud (red). (**c**) The final extracted trunk point cloud (green) after clustering process.

2.5. Global Map DT Graph Generation

A DT has multiple beneficial properties for our problem setting [47], including the fact that it maximizes the minimal angle at all the vertices of the triangulation. This means that the noise in angular values is minimal in a noisy point set. A DT also defines the so called natural neighborhood around a point (the point set connected to a given point along the triangle edges), which solves the problem of setting the local point cloud feature scope or number of neighbors in an intuitive way [44].

Our objective is to match the local trunk graph G_{local} against a subset of the global trunk graph map G_{map}, and the prerequisite is that there exist some sets of geometric structures that can be uniquely identified in both graphs, i.e., geometrically similar structures such as triangles or sets of triangles that are defined utilizing the trunks as vertices. From the trunk landmarks M_{trunks} of the PC_{map} obtained as described in Section 2.4, we get the Delaunay triangulation $M_{trunks} \rightarrow G_{map} = (V_{map}, E_{map}, T_{map})$.

Algorithm 1: Extracting trunks from the point cloud map.

```
Data: A 3D point cloud map $PC_{map} = \{p_i\}_{i=1...n_m}$
Result: 2D Trunk map $M_{trunks} = \{M_k\}_{k=1...n_t}$
foreach $p \in PC_{map}$ do
    $SearchPoint : p_{sp} = p + e_3 t_h$;
    $NearestPointSearch : p_{sn} = nearestKSearch(P_m, p_{sp})$ ;
    $ComputeDistance : distance(p_{sp}, p_{sn})$;
    if $distance < d\_threshold$ then
        $p_i.z = 0$ ;
        $P_t \leftarrow p_i$ ;
    end
end
foreach $p \in P_t$ do
    $SearchPoint : p_{sp} = p$;
    $NearestPointSearch : p_{sn} = nearestKSearch(P_t, p_{sp})$ ;
    $ComputeDistance : distance(p_{sp}, p_{sn})$;
    if $distance < d\_threshold$ then
        $PC_t \leftarrow p$ ;
    end
end
$Clusters = EculideanClusterExtraction(PC_t)$ ;
foreach $cluster \in Clusters$ do
    if $Cluster.points.size() < cluster\_threshold$ then
        Delete $cluster$ from $Clusters$
    else
        $\bar{p} = \text{mean}_{p \in Cluster}\, p$ ;
        $M_{trunks} \leftarrow \bar{p}$ ;
    end
end
```

2.6. Local Map and Point Cloud Aggregation

In a dense forest environment, nearby trunks may be blocked from the lidar view. This may result in not enough trunks which can be reliably observed also in the future locations. Thus, in our case, we will employ the LOAM method, which builds a local map from aggregating several consecutive observations. However, we will also explore the direct construction using only one frame observation.

Both the construction of the ground truth map and the aggregation of consecutive lidar frames rely on the LOAM method [33]. It operates as follows. Let $P_t = \{p_i\}_{i=1...n}$ represent a raw scanned point cloud received at time t. All these raw point cloud's are processed with the LOAM algorithm to build a local map based on one scan or the aggregation of several consecutive scans. The LOAM algorithm is a state-of-the-art feature-based lidar odometry algorithm. LOAM receives the 3D point cloud from the lidar, and projects the point cloud onto a range image for feature extraction. By calculating the curvature and some features from each row of the range image, the registration process selects subsets P_e and P_p (edge and plane points, respectively). Instead of comparing all the points, LOAM utilizes only those two subsets to find a transformation between scans, and then a two-step Levenberg–Marquardt optimization method is employed to optimize the six-degree-of-freedom transformation across consecutive scans.

The complete lidar odometry problem gets solved with a speed of 1 Hz resulting the local map PC_{local} with the computing platform utilized in our experiments.

In this study, the local point cloud PC_{local} was generated with one or several scans by utilizing the estimated position from the LOAM method. The raw lidar output frequency is 30 HZ, but not every scan from the lidar stream needs to be used to calculate the robot odometry. To reduce the accumulation error and balance the computation burden, LOAM only considers those scans that the Euclidean distance between two observation positions is longer than a certain threshold distance (e.g., 30 cm) or the angular change is larger than a certain threshold angle (e.g. 30°). As the nearby trunks or objects may block a sector of the view, aggregating more frames observation from consecutive positions helps to increase the number of the stable trunks registered into the map PC_{local}.

2.7. Local DT Graph Generation

After we have obtained the local map PC_{local}, the next step is to generate the DT graph $G_{local} = \{V_{local}, E_{local}, T_{local}\}$, just as with the global map. The methodology explained in Section 2.4 applies in this case as well, but this time producing a local map L_{local}, its 2D projected subset L_{trunks} and a local DT graph G_{local}.

2.8. Matching Local and Global DT

The DT graph obtained from the global point cloud map, $G_{map} = (V, E, T)$, defines a set V of vertices, a set $E \subseteq V^2$ of edges and a set $T \subseteq V^3$. Each vertex represents a trunk in the point cloud map. Edges establish a relation between a point and its natural neighbors it is connected within the graph. For each local DT graph that we obtain by aggregating consecutive lidar frames in real-time, our objective is to find a matching subgraph for G_{local} in the graph G_{map}. We proceed as follows in order to obtain such subgraphs.

Triangle search based on dissimilarity. A dissimilarity $d(.,.)$ of two triangles t_1 and t_2 has two properties: it is always positive, $0 \leq d(t_1, t_2)$, and zero in case of identity, $d(t_1, t_1) \equiv 0$. Typical triangle dissimilarities use an intermediate vector of two descriptors, and some sort of norm between these vectors. For instance, in [48] the authors utilize the ratio of the lengths of the shortest and longest edges, and the angle between those edges. We utilize the intermediate vector (A, l) of a triangle t with an area A and $l = L^2$, where L is the perimeter length. The vector components have the dimensionality of area (in square meters). Then, the dissimilarity $d(.,.)$ is defined as: $d(t_1, t_2) = |A_2 - A_1| + |l_2 - l_1|$.

To speed up the real-time matching process, all triangle perimeters and areas of the global DT graph have been computed offline prior to the real-time matching process. The information utilized by our algorithm while operating for real-time localization is therefore not the raw global map but instead the global DT graph that has been precomputed. In terms of comparing triangles based on the magnitude of the difference of (A, l) vectors, the triangles composed by peripheral points in a graph are usually different because of fewer observed points, so only triangles $T_{selected}$ which not include a peripheral point in the local graph G_{local} are selected. Therefore, a set of candidate triangles $\{T_{selected}\}$ are selected and used to find a matching subset in G_{map}. From $\{T_{selected}\}$, we build the corresponding graph.

There may exist multiple closely similar triangles $T_{candidate}$ in G_{map} to any specific selected triangle $T \in \{T_{selected}\}$ in G_{local}. Thus, the next step is to compare the neighboring triangles too. As we have excluded the peripheral triangles, the global subset $G_{selected}$ that will be the match of $\{T_{selected}\}$ should also have three neighbor triangles. The triangle neighborhood vote is performed using the same dissimilarity measure. As we show in Figure 6, the triangle S_0 is one of the selected triangles $S_0 \in T_{selected}$ in the local graph G_{local} and the three triangles S_1, S_2, S_3 are its neighbors.

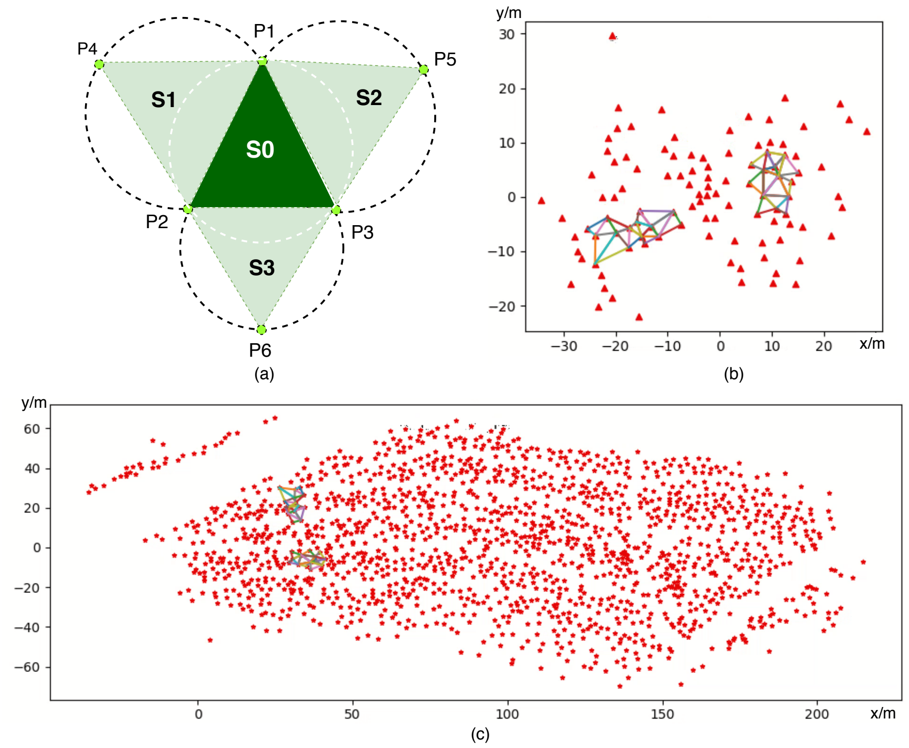

Figure 6. (**a**) The target triangle S_0 with its neighbors S_i, $i = 1.3$. (**b**) The all similar triangles found in the local trunk map, (**c**) the corresponding triangles found in global trunk map.

A feature vector of the triangle S_0 used in the match process is combined from the intermediary dissimilarity vectors of the triangle S_0 and its neighbors (called a triangle star) according to Equation (1).

$$features(S_0) = [A_0, l_0, A_1, l_1, A_2, l_2, A_3, l_3] \quad (1)$$

By comparing the $features(S_0)$ of a triangle S_0 to feature vectors in $T_{selected}$, a set of matching triangles $T_{candidate}$ meeting the error tolerance may be found. After a pair of similar triangles $(S_0, S_0') \in T_{selected} \times T_{candidate} \subset G_{local} \times G_{map}$ are found, the next step would be to estimate the position p_t and the orientation θ, which matches $T_{selected}$ with G_{map}.

Calculation of corresponding vertex pairs. In order to describe the matching process, we use the following notation. A triangle $S_0 = ABC$ consists of vertices A, B and C, and an edge vector $\vec{e} = \vec{AB}$ of an edge $e = AB$ is oriented and signified with its end points.

To solve the transformation parameters p_t and θ, we first need to find the correspondence between vertices of triangle stars $\mathcal{S} = \{S_i\}_{i=0..3} \in T_{selected}$ and $\mathcal{S}' = \{S_i'\}_{i=0..3} \in T_{candidate}$. The definition of a triangle star is easily seen from Figure 7a. As the figure shows, the vertex match between $T_{selected}$ and $T_{candidate}$ can be divided into three steps. The first step is illustrated in detail in Figure 7b. We then find the first matching pair $S_0 = ABC \in T_{selected}$ and $S_0' = MHN \in T_{candidate}$. Through comparing the side lengths between two triangles, an edge BC of the ABC can be selected so that there is an edge $|NH|$ with similar length in the MHN, so the remaining vertices A and M are a pair of corresponding vertices. For the second step, the goal is to find the other corresponding vertices in ABC and MHN. As Figure 7c shows, selecting one edge which has the vertex A as already known, and one edge contains the vertex M in the $\triangle MHN$, then we compute which side of the edge the remaining vertices C and H are located. This happens by inspecting the value $a \in \mathbb{R}$ of the following formula:

$$a = (\vec{CA} \times \vec{CB}) \cdot (\vec{HM} \times \vec{HN}) \tag{2}$$

If $a > 0$, the vertices C and H are on the same side of the edge AB and edge MN, otherwise the two vertices are on the opposite. In the case shown in Figure 7c, the result is $a < 0$, so the vertex pairs are (C, M) and (B, H). The last step is to match vertices in the triangle neighbors. Since we already know the correspondence between triangles ABC and MHN, we just need to get which two vertices are included in the neighboring triangles. For example, as we get the corresponding pairs of vertices (A, M) and (B, H) from last two steps, the last vertices E and Q are a match, too.

The match between vertices of two triangle sets is now complete. The match relation is one-to-one, so it can be recorded as a permutation function: $f : [1, 6] \subset \mathbb{N} \to [1, 6] \subset \mathbb{N}$. This way each vertex $V_i \in T_{selected}$ has a matching vertex $V_{f(i)} \in T_{candidate}$. Next we will use the permutation f while estimating the best possible translation and rotation.

2.8.1. Rotation and Translation Estimation

By comparing the vertex orientations in the found two triangle sets $\mathcal{S} = \{S_i\}_{i=0..3} \in T_{selected}$ and $\mathcal{S}' = \{S_i'\}_{i=0..3} \in T_{candidate}$, we can find a unifying 2D rigid body transformation $T[p_t, \theta]$ between the local sample and the global map. The transformation T consists of a rotation of the angle θ followed up with a translation p_t.

To calculate the transformation parameter p_t we utilize the definitions in Equation (3), where the translation t_p is the best possible one estimate, since it equates the mean of two patterns. The angle θ is defined by measuring how large a rotation is needed to transform a vectors $V_i - \overline{V}$ to $V'_{f(i)} - \overline{V}'$, where the $\overline{V}$ represents the mean of $T_{selected}$ and $\overline{V'}$ is the mean of the $T_{candidate}$.

There are six pairs $i = 1...6$ of vertices coupling $T_{selected}$, T_{local} and $T_{candidate}$ from G_{map}, and six possible candidates β_i to be chosen as the final local rotation θ. We choose the value of β_i which fits the triangle patterns $T_{selected}$ and T_{local} with the least error. One important detail is arranging a signed angular measure between two vectors. We use a rectangular rotation matrix $P = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ to get a positive $sign(Pa \cdot b)$ for rotation angles, which move a vector a to a vector b. We omit possible fit minimizing values in between the angles $\{\beta_i\}_{i=1..6}$ because the objective at this stage is to only have a close estimate and not the final position estimation.

Using a zeroth power as a shorthand when producing unit vectors $a^0 = a/\|a\|$, we define counter-clockwise oriented angles β_i, which rotate vectors V_i parallel to vectors $V_{f(i)}$. Finally, we define approximate values for the translation p_t and the rotation θ according to Equation (3).

$$\begin{cases} \overline{V} = \underset{V \in T_{selected}}{\text{mean}} \; V \\ \overline{V}' = \underset{V' \in T_{candidate}}{\text{mean}} \; V' \\ p_t = \overline{V}' - \overline{V} \\ \beta_i = sign\left(PV_i \cdot V_{f(i)}\right) \arccos\left(V_i^0 \cdot V_{f(i)}^0\right), \; i = 1..6 \\ \theta = \underset{\beta_i,\, i=1...6}{\text{argmin}} \sum_{j=1...6} \left\| \begin{bmatrix} \cos\beta_i & -\sin\beta_i \\ \sin\beta_i & \cos\beta_i \end{bmatrix} \left[V_j - \overline{V}\right]^T - \left[V_{f(j)} - \overline{V'}\right]^T \right\| \end{cases} \tag{3}$$

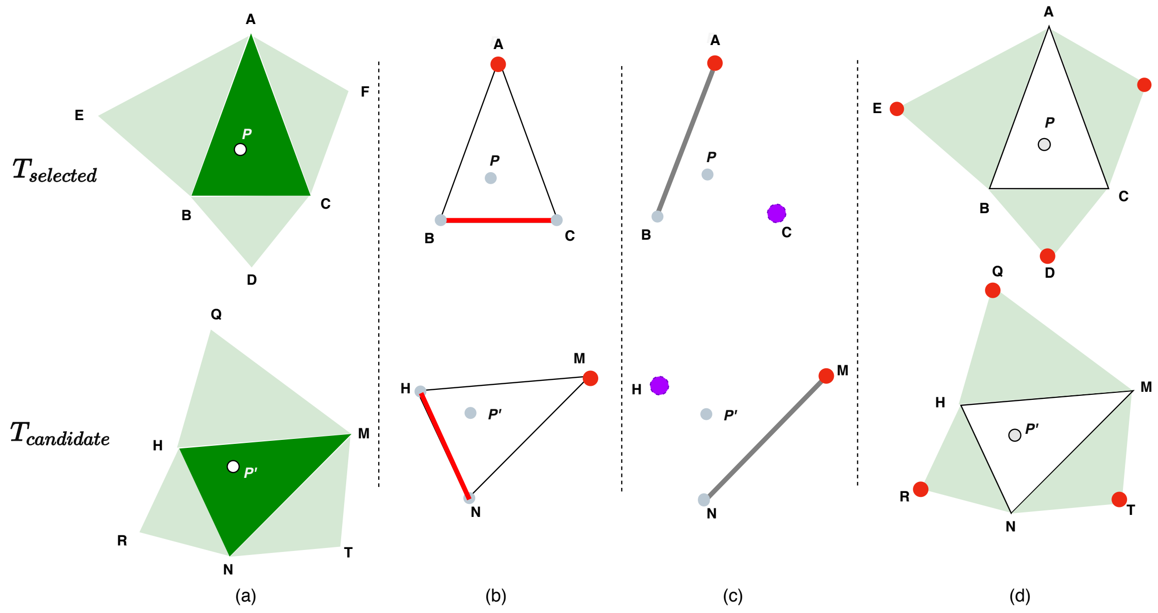

Figure 7. (**a**) The selected triangle $T_{selected}$ in G_{local} with candidate matched triangle $T_{candidate}$ in G_{map}. (**b**) Find the first corresponding point A in $T_{selected}$ with its M in $T_{candidate}$ by finding the similar featured side length. (**c**) Find the rest points B, c and its corresponding points H, N. (**d**) Find the corresponding neighbor points E, D, F and its corresponding points Q, R, T.

2.8.2. Geometric Verification, Final Translation and Rotation Estimation

From the G_{local} match against G_{map}, usually multiple matches $\{(\mathcal{S}_k, \mathcal{S}'_k)\}_{k=1...m}$ between $T_{selected}$ and $T_{candidate}$ can be found. Therefore, we need to select one among m transformation candidates with rotation and translation parameters θ_k and t_k. This final step is to find the best estimation between G_{local} and G_{map}. Let the corresponding points of a match candidate be (V_{local}, V_{map}). Starting from the previously computed initial guesses θ and p_t, we select the final solution from Equation (4). Note that this does not ensure global convergence, and unexpected results might be obtained. For instance, if the global map is too homogeneous, then different regions might have sets of trees that yield similar local maps. The risk for this can be reduced by taking into account the previous locations, or accumulating larger local maps until the matching gives a single matched subgraph from the global map with low error. Another risk is that not enough trees might be detected, which could again result in multiple global subgraphs matching with similar error. Finally, if the map is too large then multiple positions could also yield similar error. This latter scenario can be avoided by creating a set of smaller (non-disjoint) maps from the global map. Through our experiments, nonetheless, we have been able to confirm stable position estimation when enough consecutive lidar scans were aggregated.

$$(\theta, t_x, t_y) = \arg_{\beta, x_t, y_t} \min_{\substack{\beta \in B \\ x \in D_x \\ y \in D_y}} \sum_{j=1...m} \left\| \begin{bmatrix} \cos\beta & -\sin\beta & x \\ \sin\beta & \cos\beta & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_j^T - V_{f(j)}^T \end{bmatrix} \right\|, \tag{4}$$

where m is the number of local matches, $\beta \in B = [\min \theta_{j_{j=1...m}}, \max \theta_{j_{j=1...m}}] \subset \mathbb{R}$ goes through the scope occurring in the local rotation angles and x goes through a similar scope of D_x occurring in translations along the x axis. D_y is defined correspondingly on the y axis. Note that both Equations (3) and (4) require the vertices to be in a homogeneous form $V \to [V\ 1]$. The local start point is (x_0, y_0) with an orientation of θ_0, and the final estimation of the robot position is $(x_0 + t_x, y_0 + t_y)$ with an orientation θ in the global map.

3. Results

This section presents our experimental results. We analyze the performance of our method from the point of view of the trunk segmentation as well as the DT matching between the local graph and a subset of the global graph.

3.1. Trunk Segmentation Performance

Figure 8 shows the number of trunks that can be observed from a single lidar scan, and when aggregating three, five or ten consecutive frames. Figure 9 shows the number of features matched between G_{local} and G_{map} for different number of consecutive aggregated frames. In Table 1 we show the average number of trunks observed, as well as the average number of trunks that can be matched with the global map after generating the DT graph. We also include the overall matching success rate, which is defined as the number of local DT graphs that have been successfully matched with the corresponding subsets of the global DT graph. From individual lidar frames, we were only able to obtain an average of 54.09 valid trunks in total. When aggregating consecutive scans, we were able to obtain approximately a twofold increase in the number of detected trunks, and a tenfold increase in the number of matched trunks. The number of aggregated frames thus has a significant quantitative impact on the performance of our algorithm with a difference of up to an order of magnitude in the number of matched trunks. As the number of trunks and their positions determines the appearance of the local DT graph, this will directly affect the number of features detected by our descriptor. The number of trunks observed is the key factor influencing the overall results of our algorithm. From Table 1 we can see the relationship between the DT matching success and the number of trunks. With a single frame observation, only an average 1.71 trunks are successfully matched; with three aggregated frames, there are in average 4.97 trunks matched; with 5 and 10 aggregated frames, there are more than 10 trunks in average that our algorithm can match with the global map, taking the overall DT match success rate to 100% in the latter case.

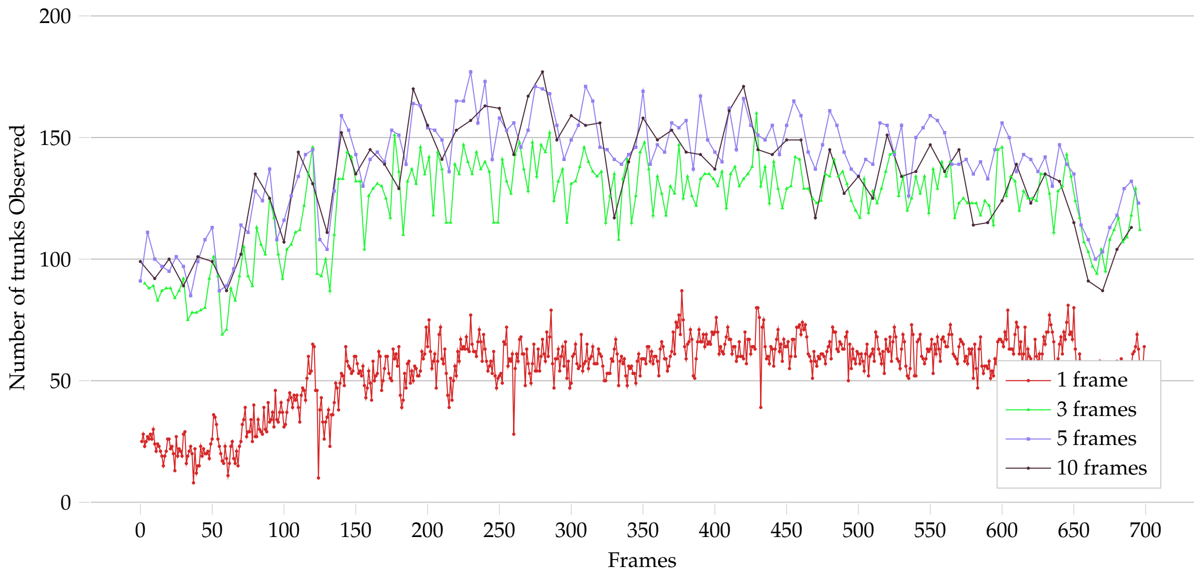

Figure 8. Number of trunks observed for different number of consecutive aggregated frames.

Table 1. Number of trunks detected and success match for different number of consecutive aggregated frames (average number of trunks matched per scan and total ratio of successful matches).

#Aggregated Frames	Average #Trunks	Average #Matching Triangles	Overall Match Success Rate
1	54.09	1.71	22.89%
3	122.98	4.97	88.79%
5	139.54	10.22	99.28%
10	133.96	14.37	100%

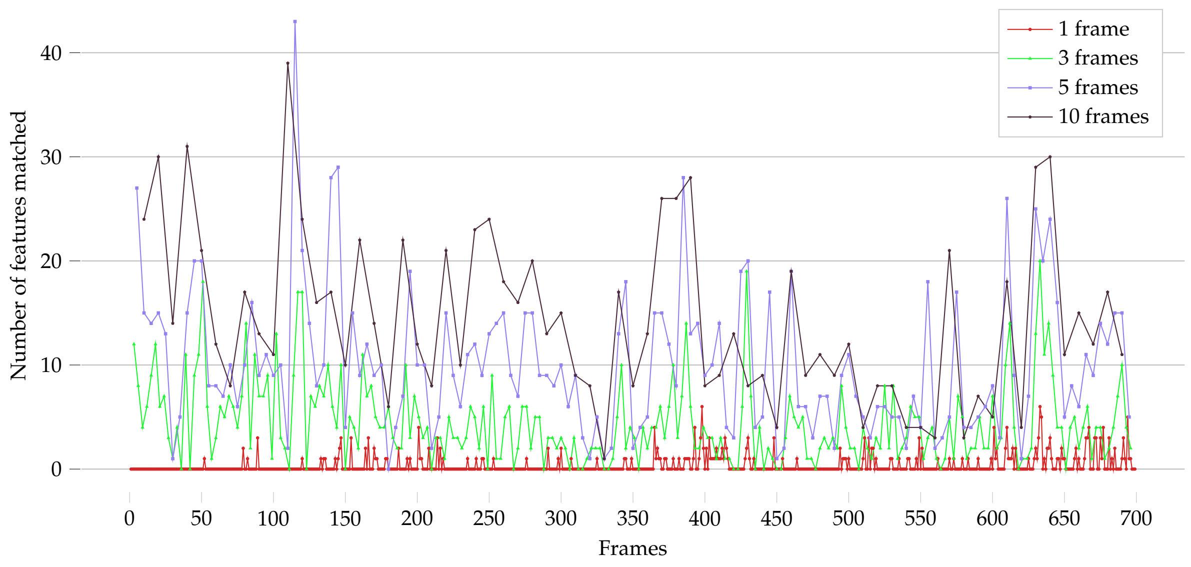

Figure 9. Number of features matched between G_{local} and G_{map} for different number of consecutive aggregated frames.

3.2. Computational Time and Accuracy

The key matching process is based on the calculation and matching of the Delaunay triangulations, which is in turn based on the aggregation of 1, 3, 5 or 10 lidar frames. These four possibilities with a varying number of aggregated frames are labeled as C1, C3, C5 and C10 in Figure 10. The aggregation of frames adds complexity and requires computing time both in the local map creation phase (black line) and the similarity matching time (blue line). We can see that the rotational location error significantly improves when aggregating more frames. The computing time of the match process grows approximately in terms of $O(\Delta_r^{-2})$, where Δ_r is the angular accuracy achieved by different choices C1, C3, C5, C10. In summary, the computation time is about 0.1 seconds for a single frame (C1) and 0.6 seconds when aggregating 10 frames (C10).

To test our pose estimation accuracy, we used the original LOAM position as the ground truth. Figure 10 depicts only the rotational error, since the translation error was approx. 0.2 m with a standard deviation of 0.14 m in all cases. It is worth noting that the rotational error is not cumulative, since the match between the local and global DT graphs is being done separately and independently at each location. The previous location is only used as the initial state. The maximum occurred translation error was approx. 0.5 m and the maximum rotation error 2.23°, both for the C1 scenario.

We can also see that the match time is relatively stable in the C1-C5 scenarios. The local map creation time (aggregation of frames) stabilizes as the number of frames increases, owing to the stabilization in the number of trees that can be observed. The segmentation algorithm is implemented in C++, while the matching process is implemented in Python.

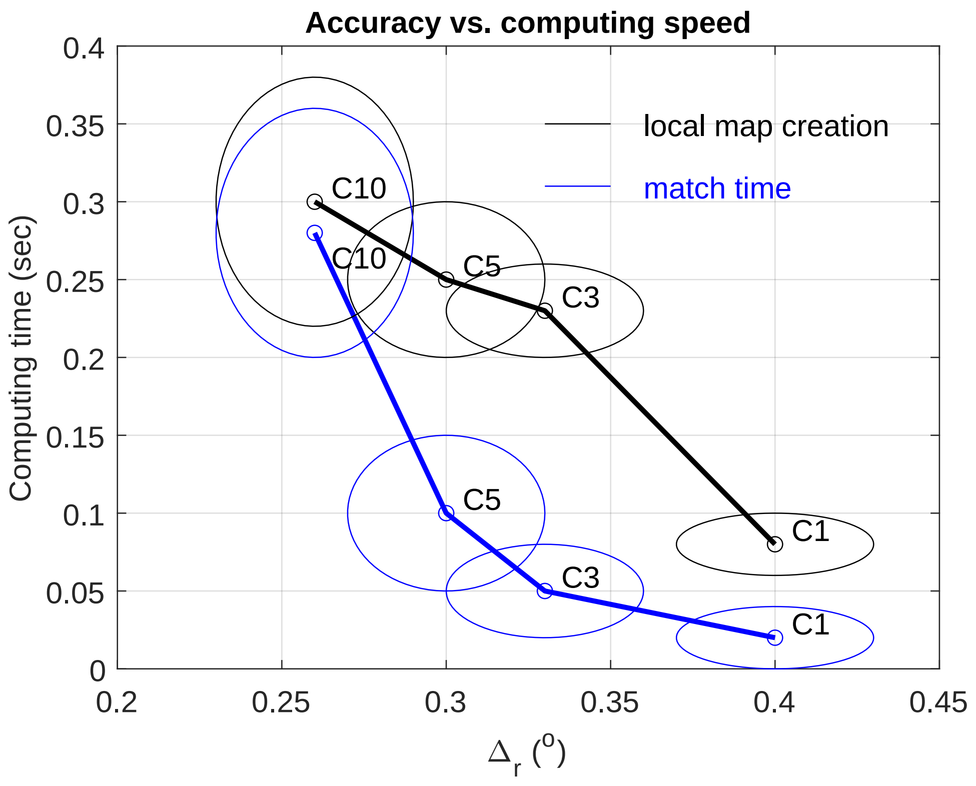

Figure 10. Dependency of the rotational accuracy and computation time on the choice of the number of aggregated frames. Both the local map creation time (black) and the similarity matching time (blue) are depicted. Number of frames ranges from 1 to 10 (C1, C3, C5, C10). Ellipses represent the 2 s.t.d. of the measurements.

4. Discussion

An application topic of the proposed navigation system is an automated transport robot, which moves logs from the forest to the collection road. Since work happens in two phases, where a forest harvester produces stripping roads and fells trees, and the collection phase, a robot can assume to have the global map built by the harvester which is not necessarily autonomous. This way its task is just to orientate itself along the strip road. A challenge arises from the high utility load of the transfer robot, which can be up to 8000 kg. A transfer robot may not have an as big capacity, but the ability to locate itself, and to detect possible stability risks is a requirement.

In addition, a transfer robot moves faster than the harvester going through its work cycle. A good estimate in rough terrain is 1.0 m/s. Thus, a fast and accurate odometric computation is essential.

4.1. Topology Mapping

In this paper, the forest trunk topology map is generated from the previous lidar data by the harvester. There are also two alternative ways to generate the global map: a large-scale (e.g., nationwide) digital forest inventory, or an aerial laser scan (ALS) from a drone working as an autonomous team member of the harvester and the robot carrier. Compared to the point cloud map, the topology map has the potential to do fast localization and dynamic update without much computation burden and makes a 3 actor teams (harvester-drone-robot) an interesting target of further study.

Even the data utilized in out experiments were gathered in a relatively flat forest stand (approximately 8 m height difference across the site), the proposed approach does not make any assumptions regarding the inclination of the terrain, but instead we assume than trees grow relatively upright. This is because the tree stem detection is aware of the vertical orientation independent of the dominant terrain orientation. There are only few relatively simple modifications to the presented algorithm, where the triangle matches and triangle stars would be defined in actual 3D. The actual effect of topographic features has to do with the reduced scope of the lidar scanning. This could be alleviated by an aided UAV unit, but our first approach is studying the usability of the tree maps in relatively benign conditions.

4.2. Potential Accuracy Improvements

As illustrated in Figure 3, each location operation is independent, except naturally each step will be initialized by the previous location to gain advantage in the corresponding search operation. In addition, one can reduce the number of localizations e.g., by assuming an interpolated movement in between sampled positions. Since this is a registration task, there is no actual error accumulation per se. The location accuracy and computation time are tightly related as seen in Figure 10. As long as the computation is fast enough, accuracy is permanently the documented one.

From the experiments, we can see that the number of trunks detected in each observation significantly influence the result of the success matching ratio. In our method, due to the lidar sensor characteristic, far trunks are unable to be recognized because of too few lidar points sampled from the appearance. Therefore, we can employ other sensors producing photogrammetric point clouds which can obtain more detail of the trunk appearance, which lets the system get more capability to recognize the trunks from PC.

Another significant source of error can be found in the trunk position estimation process. In our method, the G_{map} and G_{local} are generated from the segmented trunks points in PC_{map} and PC_{local} by calculating the average coordinate of points. However, it is impossible to get each individual trunk whole appearance observation with several consecutive lidar observations. As the observation in different positions will get different point cloud data of different sides of the trunk, the estimation pose of the trunk will be different in each local graph. However, the global graph has a more accurate trunks position as more details of each trunk collect from different sensors and different views, so the more accurate trunk position estimation in the local map can increase the chance of successful matching between local graphs and global graph. To reach a more robust system, we also can utilize other sensors like a camera through sensor fusion to get more details about each individual trunk.

4.3. Potential Computational Improvements

We proposed an efficient, robust framework to locate the robot harvester in large area forest and we tested it on a real harvester. In our case, the GNSS/GPS info is not taken into account. The location accuracy is approx. 0.3 m from consequential field measurements and approx. 2 m when comparing to general odometric result. This accuracy is enough to give a robot an initial position, and this can accelerate the matching process by conveniently initializing the search with a close match. For a real application, the GPS also can help decide which global point cloud map it will access from the cloud server for the localization.

The local Delaunay triangularization generating the local trunk map T_{local} can be implemented in a radius-limited way by using the S-Hull method [49], which limits the size n of the computational task with the well-established complexity $O(n \log n)$. A selection of an active subset of triangles can be made in the global map T_{local} based on the previous localization result. Together these improvements have the potential to make the matching process much faster.

There is a possibility to speed the convergence of the search of the final transformation in Equation (4) by using a branch-and-bound algorithm [50] with properly set estimates for local extreme. This possibility is left for future research.

All in all, the C1-C5 cases are applicable to the intended situation where the robot moves at approx. 1 m/s and has a sampling rate 2 scans/sec. The case C10 is within the reach after implementing the above-mentioned improvements.

5. Conclusions

In this study, we proposed a simple yet effective segmentation-based approach to detect trunk position and Delaunay triangulation (DT) geometry-based localization method for autonomous robots navigating in a forest environment. The proposed methods can provide accurate positioning based only on real-time lidar data processing in the unstructured and relatively complex environments that forests represent. The proposed method can be utilized for harvesters or other autonomous robots enabling fast global localization and recognizing individual trunks in real-time.

Our experiments show the proposed method reach accurate global localization precision without a good initial pose or GPS signal. The proposed method is simple and efficient, and it is a sensible solution to meet localization needs of harvesting operations in the forest. In future work, we plan to explore the forest localization algorithm in the context of significantly larger forests and to apply the proposed method at a system level for map updating or within the SLAM stack. Gathering more data, we will also be able to further analyze the performance of our algorithm when the sensors have different points of view, or when the global map is gathered in different conditions than the real-time lidar data of the forwarder, such as different vehicles or sensor settings.

Author Contributions: Conceptualization, Q.L., P.N., J.P.Q. and T.W.; methodology, Q.L., J.P.Q. and P.N.; software, Q.L.; validation, Q.L.; formal analysis, Q.L.; investigation, Q.L.; data curation, J.H., Q.L.; writing—original draft preparation, Q.L., J.P.Q. and P.N.; writing—review and editing, Q.L.; visualization, Q.L.; supervision, J.H. and T.W.; project administration, J.H.; funding acquisition, J.H. and T.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research are funded by Business Finland grant number 26004155 and Academy of Finland grant number 328755.

Acknowledgments: The data from the test site was gathered in co-operation with Stora Enso oy, Metsäteho oy, and Aalto University. The data collection was done under the EFFORTE (Efficient forestry for sustainable and cost-competitive bio-based industry (2016-2019) in WP3—Big data databases and applications.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DT	Delaunay Triangulation
SLAM	Simultaneous localization and mapping

References

1. Kankare, V.; Vauhkonen, J.; Tanhuanpaa, T.; Holopainen, M.; Vastaranta, M.; Joensuu, M.; Krooks, A.; Hyyppa, J.; Hyyppa, H.; Alho, P.; et al. Accuracy in estimation of timber assortments and stem distribution—A comparison of airborne and terrestrial laser scanning techniques. *ISPRS J. Photogramm. Remote Sens.* **2014**, *97*, 89–97. [CrossRef]
2. Hellström, T.; Lärkeryd, P.; Nordfjell, T.; Ringdahl, O. Autonomous Forest Vehicles: Historic, envisioned, and state-of-the-art. *Int. J. For. Eng.* **2009**, *20*, 31–38. [CrossRef]

3. Liao, F.; Lai, S.; Hu, Y.; Cui, J.; Wang, J.L.; Teo, R.; Lin, F. 3D motion planning for UAVs in GPS-denied unknown forest environment. In Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV), Gotenburg, Sweden, 19–22 June 2016; pp. 246–251.
4. Tian, Y.; Liu, K.; Ok, K.; Tran, L.; Allen, D.; Roy, N.; How, J.P. Search and rescue under the forest canopy using multiple UAS. In Proceedings of the International Symposium on Experimental Robotics, Buenos Aires, Argentina, 5–8 November 2018; pp. 140–152.
5. Yoneda, K.; Suganuma, N.; Yanase, R.; Aldibaja, M. Automated driving recognition technologies for adverse weather conditions. *IATSS Res.* **2019**, *43*. [CrossRef]
6. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
7. Resindra Widya, A.; Torii, A.; Okutomi, M. Structure-from-Motion using Dense CNN Features with Keypoint Relocalization. *arXiv* **2018**, arXiv:1805.03879.
8. Badue, C.; Guidolini, R.; Carneiro, R.V.; Azevedo, P.; Cardoso, V.B.; Forechi, A.; Jesus, L.; Berriel, R.; Paixão, T.; Mutz, F.; et al. Self-driving cars: A survey. *arXiv* **2019**, arXiv:1901.04407.
9. Thakur, R. Scanning LIDAR in Advanced Driver Assistance Systems and Beyond: Building a road map for next-generation LIDAR technology. *IEEE Consum. Electron. Mag.* **2016**, *5*, 48–54. [CrossRef]
10. Qingqing, L.; Peña Queralta, J.; Gia, T.N.; Zou, Z.; Westerlund, T. Multi Sensor Fusion for Navigation and Mapping in Autonomous Vehicles: Accurate Localization in Urban Environments. In Proceedings of the 9th IEEE CIS-RAM Conference, Bangkok, Thailand, 18–20 November 2019.
11. Zhang, W.; Wan, P.; Wang, T.; Cai, S.; Chen, Y.; Jin, X.; Yan, G. A novel approach for the detection of standing tree stems from plot-level terrestrial laser scanning data. *Remote Sens.* **2019**, *11*, 211. [CrossRef]
12. Pierzchała, M.; Giguére, P.; Astrup, R. Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM. *Comput. Electron. Agric.* **2018**, *145*, 217–225. [CrossRef]
13. Liang, X.; Kankare, V.; Hyyppä, J.; Wang, Y.; Kukko, A.; Haggrén, H.; Yu, X.; Kaartinen, H.; Jaakkola, A.; Guan, F.; et al. Terrestrial laser scanning in forest inventories. *ISPRS J. Photogramm. Remote Sens.* **2016**, *115*, 63–77. [CrossRef]
14. Miettinen, M.; Öhman, M.; Visala, A.; Forsman, P. Simultaneous localization and mapping for forest harvesters. In Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 517–522.
15. Tang, J.; Chen, Y.; Kukko, A.; Kaartinen, H.; Jaakkola, A.; Khoramshahi, E.; Hakala, T.; Hyyppä, J.; Holopainen, M.; Hyyppä, H. SLAM-aided stem mapping for forest inventory with small-footprint mobile LiDAR. *Forests* **2015**, *6*, 4588–4606. [CrossRef]
16. Ringdahl, O.; Lindroos, O.; Hellström, T.; Bergström, D.; Athanassiadis, D.; Nordfjell, T. Path tracking in forest terrain by an autonomous forwarder. *Scand. J. For. Res.* **2011**, *26*, 350–359. [CrossRef]
17. Zhu, X.; Kim, Y.; Minor, M.A.; Qiu, C. *Autonomous Mobile Robots in Unknown Outdoor Environments*, 1st ed.; CRC Press Inc.: Boca Raton, FL, USA, 2017.
18. Tominaga, A.; Eiji, H.; Mowshowitz, A. Development of navigation system in field robot for forest management. In Proceedings of the 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), Toyama, Japan, 5–8 December 2018; pp. 1142–1147.
19. Chen, S.W.; Nardari, G.V.; Lee, E.S.; Qu, C.; Liu, X.; Romero, R.A.F.; Kumar, V. SLOAM: Semantic lidar odometry and mapping for forest inventory. *IEEE Robot. Autom. Lett.* **2020**, *5*, 612–619. [CrossRef]
20. Sattler, T.; Leibe, B.; Kobbelt, L. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1744–1756.
21. Magnusson, M.; Nuchter, A.; Lorken, C.; Lilienthal, A.J.; Hertzberg, J. Evaluation of 3D registration reliability and speed-A comparison of ICP and NDT. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3907–3912.
22. Tomaštík, J.; Saloň, Š.; Piroh, R. Horizontal accuracy and applicability of smartphone GNSS positioning in forests. *For. Int. J. For. Res.* **2016**, *90*, 187–198. [CrossRef]

23. Zimbelman, E.G.; Keefe, R.F. Real-time positioning in logging: Effects of forest stand characteristics, topography, and line-of-sight obstructions on GNSS-RF transponder accuracy and radio signal propagation. *PLoS ONE* **2018**, *13*, e0191017. [CrossRef]
24. Rusinkiewicz, S.; Levoy, M. Efficient variants of the ICP algorithm. In Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling, Quebec City, QC, Canada, 28 May–1 June 2001; pp. 145–152.
25. Segal, A.; Haehnel, D.; Thrun, S. Generalized-icp. In Proceedings of the Robotics: Science and Systems, Seattle, WA, USA, 28 June–1 July 2009; Volume 2, p. 435.
26. Holz, D.; Ichim, A.E.; Tombari, F.; Rusu, R.B.; Behnke, S. Registration with the point cloud library: A modular framework for aligning in 3-D. *IEEE Robot. Autom. Mag.* **2015**, *22*, 110–124. [CrossRef]
27. Lauer, M.; Lange, S.; Riedmiller, M. Calculating the perfect match: An efficient and accurate approach for robot self-localization. In *Robot Soccer World Cup*; Springer: Berlin/Heidelberg, Germany, 2005.
28. Biber, P.; Straßer, W. The normal distributions transform: A new approach to laser scan matching. In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453), Las Vegas, NV, USA, 27–31 October 2003; Volume 3.
29. Yang, J.; Li, H.; Campbell, D.; Jia, Y. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 2241–2254. [CrossRef]
30. Nunez, P.; Vazquez-Martin, R.; del Toro, J.C.; Bandera, A.; Sandoval, F. Feature extraction from laser scan data based on curvature estimation for mobile robotics. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, Orlando, FL, USA, 15–19 May 2006; pp. 1167–1172.
31. Sampath, A.; Shan, J. Clustering based planar roof extraction from lidar data. In Proceedings of the American Society for Photogrammetry and Remote Sensing Annual Conference, Reno, NV, USA, 1–5 May 2006; pp. 1–6.
32. Liang, J.; Zhang, J.; Deng, K.; Liu, Z.; Shi, Q. A new power-line extraction method based on airborne LiDAR point cloud data. In Proceedings of the 2011 International Symposium on Image and Data Fusion, Tengchong, China, 9–11 August 2011; pp. 1–4.
33. Zhang, J.; Singh, S. LOAM: Lidar Odometry and Mapping in Real-time. In Proceedings of the Robotics: Science and Systems, Berkeley, CA, USA, 12–16 July 2014.
34. Shan, T.; Englot, B. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 4758–4765.
35. Thrun, S.; Koller, D.; Ghahmarani, Z.; Durrant-Whyte, H. SLAM updates require constant time. In Proceedings of the Workshop on the Algorithmic Foundations of Robotics, Nice, France, 15–17 December 2002; pp. 1–20.
36. Liu, Y.; Thrun, S. Results for outdoor-SLAM using sparse extended information filters. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), Taipei, Taiwan, 14–19 September 2003; Volume 1, pp. 1227–1233.
37. Ulrich, I.; Nourbakhsh, I. Appearance-based place recognition for topological localization. In Proceedings of the IEEE International Conference on Robotics and Automation (Cat. No. 00CH37065), San Francisco, CA, USA, 24–28 April 2000; Volume 2, pp. 1023–1029.
38. Chen, J.; Luo, C.; Krishnan, M.; Paulik, M.; Tang, Y. An enhanced dynamic Delaunay triangulation-based path planning algorithm for autonomous mobile robot navigation. In Proceedings of the Intelligent Robots and Computer Vision XXVII: Algorithms and Techniques. International Society for Optics and Photonics, San Jose, CA, USA, 18–19 January 2010; Volume 7539, p. 75390P.
39. Himstedt, M.; Frost, J.; Hellbach, S.; Böhme, H.J.; Maehle, E. Large scale place recognition in 2D LIDAR scans using geometrical landmark relations. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 5030–5035.
40. Lynen, S.; Bosse, M.; Siegwart, R. Trajectory-based place-recognition for efficient large scale localization. *Int. J. Comput. Vis.* **2017**, *124*, 49–64. [CrossRef]
41. Bosse, M.; Zlot, R. Place recognition using keypoint voting in large 3D lidar datasets. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 2677–2684.

42. Bosse, M.; Zlot, R. Keypoint design and evaluation for place recognition in 2D lidar maps. *Robot. Auton. Syst.* **2009**, *57*, 1211–1224. [CrossRef]
43. Qian, C.; Liu, H.; Tang, J.; Chen, Y.; Kaartinen, H.; Kukko, A.; Zhu, L.; Liang, X.; Chen, L.; Hyyppä, J. An integrated GNSS/INS/LiDAR-SLAM positioning method for highly accurate forest stem mapping. *Remote Sens.* **2017**, *9*, 3. [CrossRef]
44. Edelsbrunner, H. Triangulations and meshes in computational geometry. *Acta Numer.* **2000**, *9*, 133–213. [CrossRef]
45. Bentley, J.L. Multidimensional binary search trees used for associative searching. *Commun. ACM* **1975**, *18*, 509–517. [CrossRef]
46. Rusu, R.B. Semantic 3d object maps for everyday manipulation in human living environments. *KI Künstliche Intell.* **2010**, *24*, 345–348. [CrossRef]
47. Lee, D.T.; Schachter, B.J. Two algorithms for constructing a Delaunay triangulation. *Int. J. Comput. Inf. Sci.* **1980**, *9*, 219–242. [CrossRef]
48. Arzoumanian, Z.; Holmberg, J.; Norman, B. An astronomical pattern-matching algorithm for computer-aided identification of whale sharks Rhincodon typus. *J. Appl. Ecol.* **2005**, *42*, 999–1011. [CrossRef]
49. Sinclair, D. S-hull: A fast radial sweep-hull routine for Delaunay triangulation. *arXiv* **2016**, arXiv:1604.01428.
50. Boukouvala, F.; Misener, R.; Floudas, C. Global Optimization Advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO. *Eur. J. Oper. Res.* **2015**, *252*. [CrossRef]