Improving kinematic laser scanning point cloud accuracy with graph optimization

Jarkko Jalovaara

School of Science

Bachelor's thesis
Espoo 11.10.2021

Supervisor

Assoc. Prof. Pauliina Ilmonen

Advisor

DSc (Tech.) Antero Kukko

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the bachelor's thesis

Author Jarkko Jalovaara

Title Improving kinematic laser scanning point cloud accuracy with graph optimization

Degree programme Engineering Physics and Mathematics

Major Mathematics and Systems Sciences **Code of major** SCI3029

Teacher in charge Assoc. Prof. Pauliina Ilmonen

Advisor DSc (Tech.) Antero Kukko

Date 11.10.2021 **Number of pages** 23+2 **Language** English

Abstract

Contemporary forest management relies on accurate forest inventory data with information on individual tree level. There is a need for efficient and automated data collection in forests since collecting tree-specific data from forests manually is often laborious and time consuming. One of the most efficient tools for forest data collection is laser scanning that creates accurate 3D visualization of forest environment. However, the use of laser scanning in environments where GPS signal is lost or weak creates severe inaccuracies in the resulting 3D visualization.

The goal of this thesis is to use a Simultaneous Localization and Mapping - based framework to improve the accuracy of a laser scanning visualization in a forest environment. We use automated tree detection for point cloud data to create constraints for a least-squares minimization problem presented as a graph. We compare the results of the framework in different forest plots and determine the feasibility of the framework using an automated tree detection algorithm.

The results show that using this framework, the accuracy of the landmark locations in a point cloud is increased consistently from decimeter level to millimeter level. Results also confirm that the overall accuracy of the point cloud doesn't necessarily increase as fast as the landmark locations accuracy. This is most notable with point cloud data that initially contains significant height errors from GPS-deterioration. Overall, the framework works as intended while there is still room for further development, especially with erroneous height data or problems with ground detection.

Keywords Graph optimization, Simultaneous Localization and Mapping, Laser scanning, Least squares minimization, Forest inventory

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Tekniikan kandidaatintyön tiivistelmä

Tekijä Jarkko Jalovaara

Työn nimi Improving kinematic laser scanning point cloud accuracy with graph optimization

Koulutusohjelma Teknillinen fysiikka ja matematiikka

Pääaine Matematiikka ja systeemitieteet **Pääaineen koodi** SCI3029

Vastuuopettaja ja ohjaaja Professori Pauliina Ilmonen, TkT Antero Kukko

Päivämäärä 11.10.2021 **Sivumäärä** 23+2 **Kieli** Englanti

Tiivistelmä

Nykyaikainen metsänhoito hyödyntää tarkkaa metsäkohtaista aineistoa, jossa on tietoa yksittäisten puiden ominaisuuksista. Koska aineiston kerääminen yksittäisistä puista on usein työlästä ja aikaavievää, on tarvetta tehokkaalle ja automatisoidulle menetelmälle aineiston keräämiseen. Yksi tehokkaimmista menetelmistä aineiston keräämiselle on laserkeilaus, jolla voidaan luoda kolmiulotteinen malli metsäympäristöstä. Laserkeilauksen käyttö ympäristössä, jossa GPS-signaali on heikko tai kokonaan hävinnyt, aiheuttaa kuitenkin selkeitä epätarkkuuksia kolmiulotteisessa mallissa.

Tämän kandidaatintyön tavoitteena on käyttää Simultaneous Localization and Mapping -pohjaista menetelmää laserkeilauksella saadun kolmiulotteisen mallin tarkuuden parantamiseksi metsäympäristössä. Työssä käytetään automaattista puuntunnistusta pistepilviaineistolle pienimmän neliösumman menetelmän rajoitteiden luomiseksi graafimuodossa. Työssä verrataan menetelmän tuloksia eri metsäpalstoilla ja määritetään menetelmän toimintavarmuus automaattisella puuntunnistuksella.

Tulokset osoittavat, että menetelmää käyttämällä pistepilvimallin puiden sijaintien tarkkuus paranee johdonmukaisesti desimetritasolta millimetritasolle. Tuloksista nähdään myös, ettei pistepilven yleinen tarkkuus välttämättä parane puiden sijaintien tarkkuuden mukana. Ilmiö huomataan kaikista selkeimmin sellaisilla pistepilvimalleilla, joissa on selkeitä korkeussunntaisia virheitä GPS-signaalin puuttumisen takia. Yleisesti menetelmä toimii kuten haluttua, mutta jatkokehitykselle on edelleen tarvetta varsinkin virheellisen korkeusdatan ja puutteellisen maanpinnantunnistuksen kohdalla.

Avainsanat Graafioptimointi, Pistepilvi, Laserkeilaus, Pienimmän neliösumman menetelmä, Metsänhoito

Contents

1 Introduction

Forests supply a wide range of products and ecosystem services that are beneficial from ecological, economical as well as social standpoint. Some examples of these services include timber, biofuel, recreation services and habitat for species to support biodiversity. Forest inventory aims to provide essential reference data from forests to support all decision-making in forest ecosystems ranging from harvest planning to landscape-related arrangements.(Hyyppä et al., 2020)

During the past 10 years, one of the most widely used remote sensing technologies for forest inventory data collection has been the airborne LiDAR laser scanning method. (Pierzchala et al., 2018) Laser scanner uses LiDAR to create range estimates from the surrounding objects and GPS-based location data to create point cloud of the environment in order to visualize it in 3D. Several laser scanning methods and equipment have been deployed such as terrestrial laser scanning with stationary scanners, over-canopy-aircraft laser scanning and more recently mobile laser scanning with hand-held, backpack-portable or UAV-operated laser scanners.(Hyyppä et al., 2017)

In this thesis, a graph optimization based Simultaneous Localization and Mapping algorithm (SLAM) is used to improve the accuracy of an existing 3D point cloud visualization in forest environment. This point cloud data is gathered using a mobile backpack laser scanner combined with Global Navigation Satellite System – Inertial Measurement Unit (GNSS-IMU) -localization equipment to capture the original trajectory of the scan. Since backpack scanning is performed under-canopy, the deterioration of GPS signal results in a trajectory drift (Kukko et al., 2017) which will be corrected using SLAM. To determine the accuracy improvement, error statistics from both the original and corrected point cloud are compared. Point clouds are gathered in forest environment and only the point cloud trajectory is optimized to make the optimization problem computationally easier. After the trajectory is optimized, the original laser scanner data is combined with the optimized trajectory information for better accuracy.

The goal of this thesis is to find out how much improvement the Graph-SLAM framework gives to an existing point cloud data, identify the main difficulties with the algorithm at its current state, and analyze the effects of different parameters and environmental attributes in the performance of the algorithm. Key question of this thesis is to find out whether Graph-SLAM corrected 3D point cloud data can be used to possibly extract individual tree attributes to support decision making. Additionally, this thesis aims to briefly answer how much different environmental parameters affect the optimization process and how to potentially improve the process in the future.

The structure of the thesis is the following. In Section 2, the summary of mobile laser scanning methods and data sources are presented. In Section 3, the theoretical framework of graph optimization and Graph-SLAM to correct the trajectory GPS data is presented. Section 4 presents the results of the optimization process and error statistics comparison. Section 5 completes the work with overall summary.

2 Background

2.1 Forest Inventory

Accurate forest inventory data is essential to prevent wrong decisions in forestry. Individual tree attributes, such as tree height and DBH (diameter at breast height) are especially important since estimates of individual tree volume, for example, can be derived from these values. Collecting data from every single tree of the forests separately for large plots is, however, generally not a feasible option since individual measurement of trees is labour intensive and especially time consuming. (Jaakkola et al., 2017; Hyyppä et al., 2020)

In an attempt to make forest data collection less laborious and more efficient, data collection technologies based on remote-sensing have been studied.

The most important technology in last decades has been LiDAR (Light Detection and Ranging), which is sometimes referred to as 3D laser scanning. The main idea behind the LiDAR technology is to create an accurate 3D environment visualization in a point cloud format. In LiDAR, the laser scanner emits laser beams around the scanner and measures the time between the laser emission and the return time of the laser after deflection from some surrounding objects. This results in a set of points in 3D space representing the location of the objects relative to the scanner, namely a 3D point cloud. To create a full 3D map from the environment, as in Picture 1, separate point clouds are combined together in a global coordinate system by assigning a global GPS-location and scanner orientation for every separate point cloud. This can be done with integrated GNSS-IMU -system which collects GPS location and orientation information of the scanner. (Balenović et al., 2021; Næsset et al., 2004)

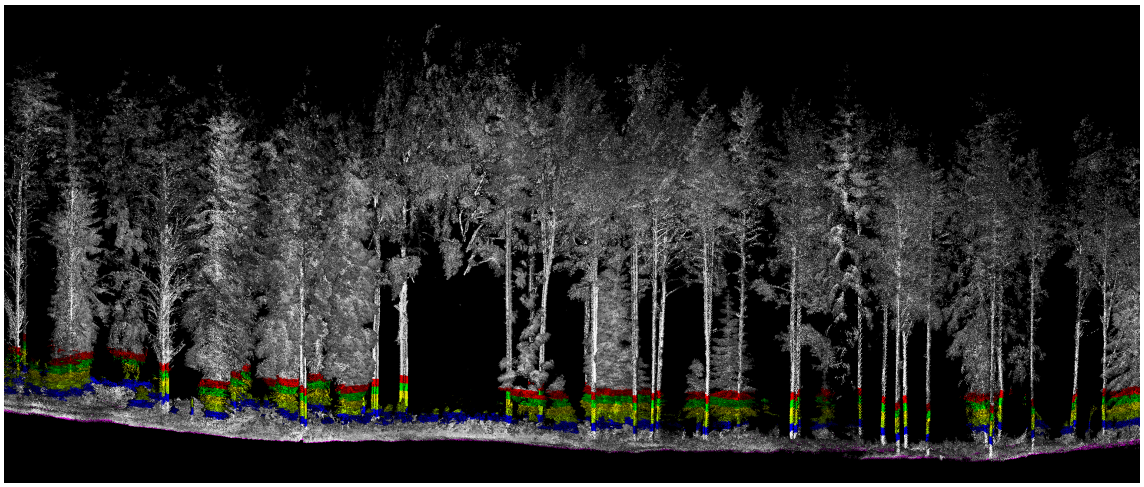

Figure 1: An example of a 3D map cross section of a forest in a point cloud format. The colored bars near the ground are colored height intervals of the point cloud. The empty area in the middle of the picture is the blind spot of the laser scanner, which can be interpreted as part of the scanner trajectory. The picture was processed by Antero Kukko.

2.2 Mobile laser scanning

In comparison to Terrestrial based Laser Scanning TLS or conventional field measurements, the mobile laser scanning methods are significantly faster for collecting data to extract individual tree attributes. (Hyyppä et al., 2020) However, the accuracy of mobile laser scanning is currently not sufficient to provide reliable data for tree specific attributes. The primary source of data accuracy reduction is the weak or absent GPS signal in dense forest environments that creates trajectory drift. Merely the odometry measurements from IMU are generally not accurate enough to prevent the trajectory from drifting over longer periods of time. Without reliable positional accuracy of the laser scanner, combining individual measurements into a single point cloud becomes inconvenient decreasing the overall accuracy of a complete point cloud. (Kukko et al., 2017; Chang et al., 2019)

In our case, the laser scanner along with the GNSS/IMU localization system is mounted on a backpack (Picture 2) moving around the environment with the user. Due to dense canopy, the GPS signal can deteriorate or even become absent creating errors in GPS-location data for point clouds. The resulting 3D map accuracy therefore suffers with the loss of GPS signal, potentially creating duplicates of a separate trees in the point cloud.

In this work, the goal is to use a modified graph-SLAM algorithm to reduce the error in 3D environment maps in point cloud format. Based on the interdependencies of observations and the initial GPS-trajectory, the graph-SLAM algorithm creates a new trajectory with improved accuracy. Using the improved GPS-trajectory to combine the original laser scan measurements into a single point cloud, we can expect an accuracy improvement in a new 3D environment point map (Kukko et al., 2017).

In the paper (Kukko et al., 2017) this trajectory correction framework in forest environment was tested with three independent forest plot data sets. The results show that after the use of Graph-SLAM, the point cloud accuracy was significantly improved. The distance STD from the reference tree locations to corrected point cloud tree locations was roughly 7 millimeters, where as without trajectory correction the same value was roughly 127 millimeters. We are exploring, whether similar results can be achieved in different forest environment with slightly modified version of the algorithm.

3 Data review and Methods

3.1 Point cloud and trajectory data

The two primary datatypes used in the work are point cloud data and trajectory data. Point cloud data gathered using backpack laser scanner consists of a group of points located in a global 3D-space associated with some information for each point. In our case, the only needed information for each point are their respective GPS timestamps given in seconds for ongoing week as shown in Table 1. As we later notice, this information is vital for the graph-SLAM algorithm initialization.

Similarly, the trajectory data consist a set of points in 3D-space associated with

Figure 2: The backpack mounted laser scanner and GNSS/IMU -equipment for simultaneous trajectory data and point cloud data collection. The picture was taken by Antero Kukko.

Table 1: The different data types used in this work and their components and units for each data point. Here STD refers to standard deviation of the component

Point cloud data	**Trajectory data**
GPS Timestamp (s)	GPS Timestamp (s)
X-coordinate (m)	X-coordinate (m) and STD
Y-coordinate (m)	Y-coordinate (m) and STD
Z-coordinate (m)	Z-coordinate (m) and STD
	X-axis rotation (°) and STD
	Y-axis rotation (°) and STD
	Z-axis rotation (°) and STD

extra information. This data is gathered using specialized GNSS-IMU -system that uses GPS signals and odometric technologies to capture the trajectory of the laser scanner. Information for each point is shown in Table 1. In this work we use the orientation data, 3D location data, associated GPS timestamps and also the standard deviation estimates for both the location and the orientation. These are also required for the graph-SLAM initialization. The orientation data combined with the location data is commonly referred to as pose data.

It is important to note that during every initial scan of the laser scanner, the point clouds are in the topocentric coordinate system, that is, the origin of the system is the laser scanner itself. To create the complete point cloud data used in our work, a Helmert Transformation is used to first transform the coordinates to the geocentric coordinate system, where the origin is in the mass-center of Earth. This operation is commonly referred to as georeferencing and it involves performing the transformation to every point in the point cloud. The general Helmert Transformation works as follows:

$$\boldsymbol{R'} = \mu R(\boldsymbol{R} - \boldsymbol{R_0}), \tag{1}$$

where $\boldsymbol{R'}$ is a single point in a point cloud transformed into new coordinate system, μ is the scaling factor, R is a rotation matrix containing rotations around all 3 cartesian coordinate axes, $\boldsymbol{R}$ is the original point and $\boldsymbol{R_0}$ represents the new origin coordinates given in the old coordinate system. Note that in equation (1) the points are given as a set of coordinates in $\mathbb{R}^3$

$$\boldsymbol{R'} = \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}, \boldsymbol{R} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \boldsymbol{R_0} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

as column vectors.

In our case, the xy-coordinates are represented after the transformation as coordinates in the Mercator map projection and z-coordinate is represented as distance from the sea level. (Vermeer, 2019)

3.2 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) algorithms aim to keep track of an object's location and orientation (commonly referred to as pose) while updating a map around the moving object in real time. SLAM-algorithms are widely used, for example, in autonomous vehicles such as self-driving cars, in drones as well as in robotics, primarily as a core technique for autonomous robot design. (Kumar, 2020; Durrant-Whyte and Bailey, 2006) More formally, the SLAM-algorithms aim to solve a following probabilistic problem given the measurement data from object's sensors z and object's odometry measurements u at some discrete time $t \in [1, T]$ with an initial guess x_0

$$p(x_{1:T}, \boldsymbol{m} \mid z_{1:T}, u_{1:T}, x_0), \tag{2}$$

where x_t represent the object's true pose at time t and $\boldsymbol{m}$ is the map of the environment consisting a set of landmark attributes. Note that the general notation $y_{1:T}$ refers to a set of time indexed points from 1 to T and that all the posterior data is used in solving the problem. An odometry measurement u_t refers to relative information between consecutive robot poses x_t and x_{t+1}. In our case, it suffices to find the set of laser scanner poses $x_{1:T}$ ie. trajectory and the environment map $\boldsymbol{m}$ of the highest probability instead of the whole posterior probability distribution. Different methods of solving this SLAM problem include statistical methods such as Kalman filter as well as Monte Carlo -based particle filtering methods. (Stachniss et al., 2016; Thrun and Montemerlo, 2006)

In this work, the SLAM-algorithm is not used to create maps and backpack scanner trajectories in real time. Instead, the existing map and trajectory are post-processed to create an improved GPS-trajectory. Using the improved trajectory, we are essentially assigning new GPS-location and orientation values for separate 2D point cloud measurements, which improves the overall point cloud accuracy when separate 2D measurements are combined together as presented in Chapter 3.1.

The key assumption in most of the SLAM solving methods is that both the sensor data $z_{1:T}$ and odometry measurements $u_{1:T}$ are equally affected by the Gaussian noise. If the measurements were not affected by any noise, the trivial solution for the SLAM-problem poses $x_{1:T}$ would be to define a starting pose x_0 and then obtain the rest of the poses using exact velocity information from each point recursively (Grisetti et al., 2017). Another assumption is that all sensor data and odometry measurements depend only on the relative pose between the object and the observed landmark (sensor data) or the following object's location (odometry measurement).

It can be shown that under these assumptions the most probable set of variables $x_{1:T}$ and $\boldsymbol{m}$ in Equation (2) can be found by minimizing a sum of error functions that are derived from a graph representation of a SLAM-problem. This method is better known as Graph-SLAM. (Grisetti et al., 2011)

3.3 Forest Graph-SLAM

The Graph-SLAM is a type of SLAM-algorithm that first creates a factor graph which is then optimized according to certain constraints. The Forest graph-SLAM algorithm suitable for trajectory corrections in forest environment can be divided into three steps:

- Estimate tree trunk locations from the laser scanning data.
- Create a graph from tree trunk locations and original trajectory.
- Optimize the graph with numerical Gauss-Newton algorithm for nonlinear least-squares problem.

Tree trunk locations in $\mathbb{R}^3$ are estimated from the original point cloud data. The idea is to first extract the ground level height of the terrain from the point cloud data. After ground level extraction, the point cloud is filtered to contain only points

with a z-coordinate between 3-3.5 meters from the extracted point cloud. Trunks are identified from the filtered point clouds by finding arc-like forms from the point cloud using fitting. Finally, the arcs from the separate filtered point clouds are clustered together creating the ultimate location estimates for the tree trunks in $\mathbb{R}^3$ global coordinates (Picture 3). Additionally, for every tree location in $\mathbb{R}^3$, we collect the GPS timestamp of the observation (median of trunk points' timestamps), trunk radius estimate (average of detected arcs' radius estimates) and number of observed points representing a certain tree trunk.

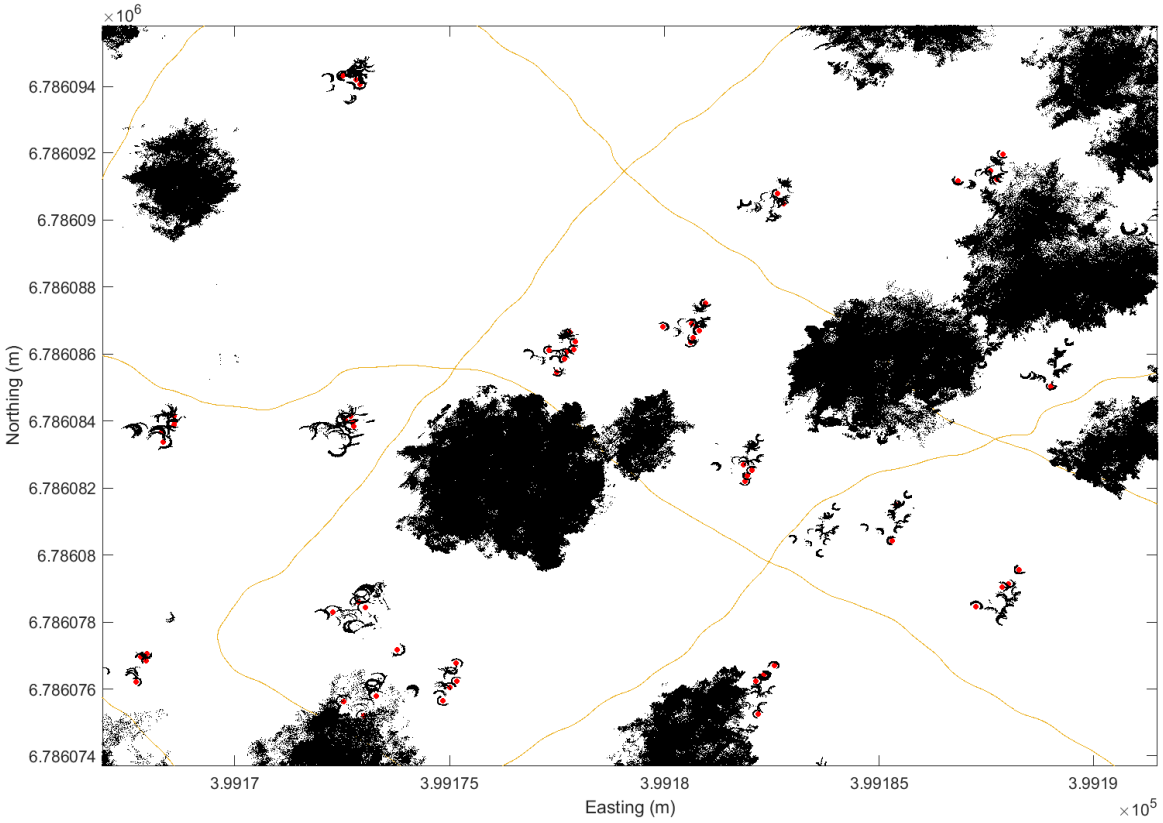

Figure 3: Example of tree detection in Plot 1 point cloud environment (top view) Black represents points in the point cloud with a z-coordinate between 3-3.5 meters. Red represents detected tree trunks from the point cloud with arc detection and orange line represents laser scanner trajectory. The multiple clustered arc-like figures can be considered to be errors in point cloud due to GPS-deterioration and the respective arc detections are assumed to be from the same tree.

After tree locations estimation, the optimizable graph is created. This graph contains 2 types of nodes and edges between them. The nodes $\boldsymbol{x}$ in the graph $C(\boldsymbol{x}, \boldsymbol{z})$ are either poses of the laser scanner (initially from the laser scanner trajectory) or tree trunk locations (initially from the previous step). The edges between the nodes are measurements $\boldsymbol{z}$ from the scanner: The edges between separate poses are odometry

measurements from the GNSS/IMU system and the edges between tree locations and poses are sensor data measurements created by comparing tree observations from point cloud and location information from a single pose. (Kukko et al., 2017) An example of an optimizable graph is presented in Picture 4.

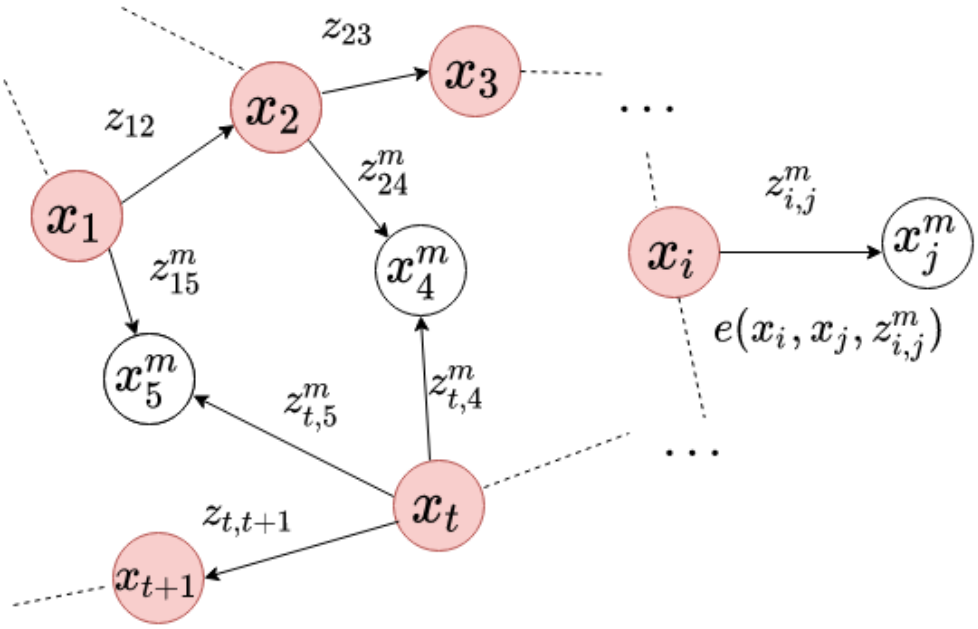

Figure 4: An example of an optimizable graph that is used for creating the objective function. For clarity, the tree observations are labeled white and marked with upper index m while scanner poses are labeled red. Note that edges between tree observations and scanner poses take into account only location data while the edges between consecutive poses use both location and orientation data.

3.4 Graph optimization

The core idea of the graph-optimization is presented next. To obtain the optimized trajectory for the laser scanner, we want to find a set of nodes $\boldsymbol{x} = (\boldsymbol{t}, \boldsymbol{q})$ that best support the odometry measurements and sensor data from the separate point clouds. The poses contain the 3-dimensional location of the scanner $t_i \in \mathbb{R}^3$ and the orientation described with rotation axis $r'_i \in \mathbb{R}^3$ and rotation angle θ_i as unit quaternion $q_i = Q(\theta_i, r'_i)$. For basic preliminaries of quaternions, see Appendix A. Decision variables in our optimization problem are the graph nodes, that is the tree locations and the laser scanner poses. The edges, on the other hand, can be considered the constraints of the problem: The edges z_{ij} between nodes x_i and x_j are the measured relative transformations between the poses. For edges between scanner pose and tree location marked as $z^m_i{}_j$, we consider only 3D locational data, thus for these edges $z^m_i{}_j = t_{ij}$. For the edges between the consecutive scanner poses, the orientation data is also used: $z_{ij} = (t_{ij}, q_{ij})$. The difference between the measured transformation and the actual transformation between the two nodes is described with the error function $e(x_i, x_j, z_{ij})$. (Grisetti et al., 2010)

The optimized trajectory (along with possibly corrected tree locations) is thus obtained by minimizing the sum of all error function values in the whole graph. Visualization of the decision variables prior and after the optimization process is shown in Picture 5. The complete optimization problem can be presented as

$$F(\boldsymbol{x}) = \sum_{i,j \in C} e(x_i, x_j, z_{ij})^T \Omega_{ij}(x_i, x_j, z_{ij}) \tag{3}$$

$$\boldsymbol{x}^* = \operatorname{argmin} F(\boldsymbol{x}),$$

where $e(x_i, x_j, z_{ij})$ is the column vector containing the error values from location part of constraint and orientation part of the constraint. The $\Omega_{ij}(x_i, x_j, z_{ij})$ for certain constraint between two nodes is the entry in the information matrix which tells us about the accuracy of the certain edge. The estimated accuracy information of the constraint z_{ij} is the inverse of covariance of the measurement data, which is obtained using standard deviation estimates of input data from GNSS-IMU - system as explained in Section 3.1. The information estimate for single constraint can be considered an error function weight for certain edge, where errors with higher measurement accuracy are weighted more and vice versa. Ultimately the measurements with higher covariance (and therefore higher possible measurement error) decrease the weight of the error function and its overall cost in the system. (Jelineck, 2016)

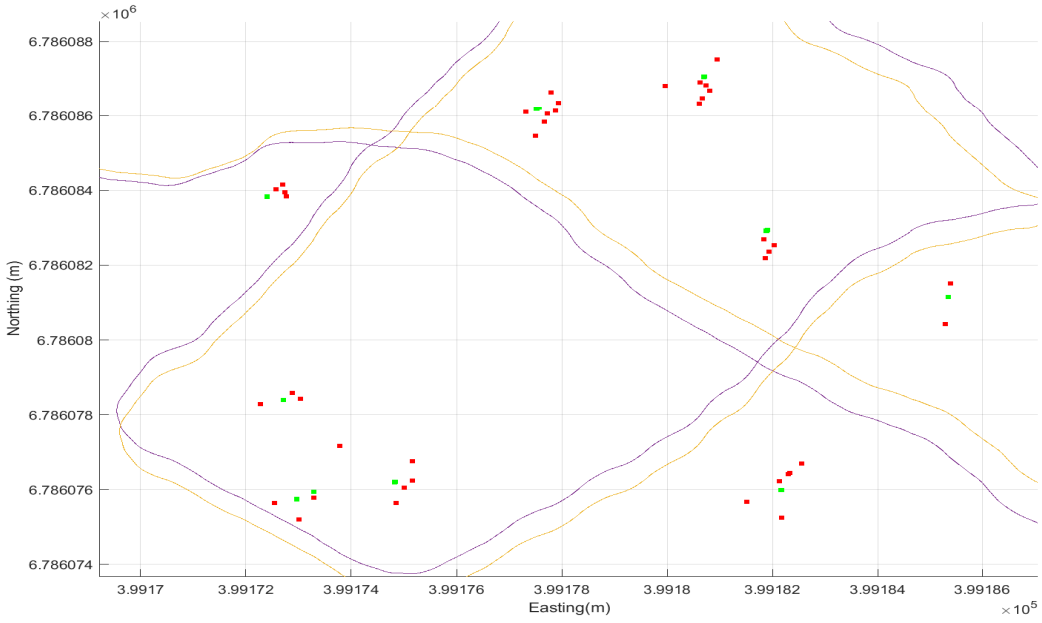

Figure 5: Visualization of some of the decision variables $\boldsymbol{x}$ prior and after the optimization in forest sample Plot 1. Yellow line shows the initial trajectory and red points show the initial observations. Purple line shows the optimized trajectory and green points show the optimized observations.

3.5 Error Functions

Graph-optimization is not only used in solving SLAM-problems. Many other problems such as Bundle Adjustment use similar graph representation to derive the objective function. The general graph-optimization framework g2o uses the sum of error functions to create the objective function for the minimization problem and can be used for BA problems and SLAM-problems alike. Consequently, this graph-optimization framework is not sensitive to sensor data formats, that is, any type of measurable physical quantity can be used as a base for graph-optimization. This means that the error function $e(x_i, x_j, z_{ij})$ must be created separately for each specific problem. (Grisetti et al., 2011, 2017)

In Forest Graph-SLAM the goal is to compare the measured relative transformations of both the location data $t_i \in \mathbb{R}^3$ as well as orientation data of the scanner $q_i = Q(\theta_i, r_i')$. Generally the error function between 2 decision variables in g2o graph-optimization framework can be written as

$$e(x_i, x_j, z_{ij}) = |h_k(x_i, x_j) - z_{ij}|, \tag{4}$$

where z_{ij} is the measured transformation (edge) and h_k is the synthetic measurement function that calculates the actual transformation between nodes x_i and x_j.

For example, the error function between tree observations and scanner poses is easy to determine since only 3-dimensional location data is considered. According to equation (4) The transformation between two points in $\mathbb{R}^3$ is determined by translation vector between the points and therefore the error for these edges is simply

$$e(x_i, x_j, z_{ij}) = |t_{ij} - t_i - t_j|, t \in \mathbb{R}^3, \tag{5}$$

that is the difference between measured location transformation and actual location transformation between x_i and x_j.

The error function between consecutive poses uses 3-dimensional location data as well as orientaton data. This error function value is thus a 2-dimensional column vector consisting of the location transformation error between two points as described in equation (5) and orientation transformation error described using unit quaternions. The difference between measured orientation and actual orientation normally is simply $q_{ij}(q_j q_i^{-1})^{-1}$, where $q_i = Q(\theta_i, r_i')$. Orientation information, however, is also used to define estimated orientation transformations in this case. The overall error function for consecutive poses is

$$e(x_i, x_j, z_{ij}) = (z_{ij} \oplus (x_i^{-1} \oplus x_j))_{[1:6]},$$

where $\oplus$ is defined as motion composition operator

$$x_i \oplus x_j = \begin{bmatrix} q_i(t_j) \\ q_i \cdot q_j \end{bmatrix},$$

such that $(\cdot)_{[1:6]}$ is an operator selecting first 6 elements of the vector argument. (Grisetti et al., 2010)

3.6 Least-Squares Solving method

The objective function (3) minimization can be carried out using Gauss-Newton algorithm for nonlinear least-squares problems. This iterative algorithm uses local Taylor-approksimations around the current trajectory estimate to create increments that minimize the value of the objective function. For simplicity of notation we define $e(x_i, x_j, z_{ij}) := e_{ij}(\hat{\boldsymbol{x}})$ The framework of single iteration in the algorithm is as follows:

- Create Taylor expansions of the error functions around the current estimate of $\boldsymbol{x}$ using Jacobian J_{ij} of e_{ij}
- Create the objective function local approksimation with the error function expansion and reformulate the objective function in quadratic form.
- Compute the increment $\boldsymbol{x}'$ by solving the linear system of equations
- Obtain solution by adding the increment $\boldsymbol{x}'$ to the current estimate $\boldsymbol{x}$

Assuming all the components of the decision variables $\boldsymbol{x}$ are euclidean, it suffices to create Taylor expansion of the error function normally as $e_{ij}(\hat{\boldsymbol{x}} + \Delta\hat{\boldsymbol{x}}) \simeq e_{ij} + J_{ij}\Delta\hat{\boldsymbol{x}}$. In our case, however, the orientation data $q_i = Q(\theta_i, r_i')$ is described with unit quaternion to prevent singularities in the optimization process. The components in this representation span over the non-euclidean rotation group $SO(3)$, which prevents the use of ordinary Gauss-Newton solving method.

The restriction to use only euclidean components in the optimization can be avoided by interpreting the SO(3) space as a manifold, a space which behaves locally as an euclidean space. The workaround is to substitute the normal + -operator with $\boxplus$ -operator that serves as a map for variation $\Delta\boldsymbol{x}$ from the Euclidean space to a manifold. More formally, the $\boxplus$ -operator is a map

$$\boxplus : S \times \mathbb{R}^3 \to S$$

that takes current state space S and small variation in $\mathbb{R}^3$ and yields the resulting state space in S. In our case the state space consists of location in $\mathbb{R}^3$ and orientation in $SO(3)$. Variance in euclidean location can be easily expressed by simple vector addition, whereas variation in orientation is expressed using unit quaternions q. The formulation of $\boxplus$ for our state variable $\boldsymbol{x}$ is

$$\boldsymbol{x} \boxplus \Delta\boldsymbol{x} = \begin{bmatrix} t + \Delta t \\ q \cdot exp(\frac{\Delta q}{2}) \end{bmatrix},$$

where q is unit quaternion and t is location in $\mathbb{R}^3$ Hertzberg et al. (2013).

The Taylor expansion of the error function in the manifold environment can be presented as follows:

$$e_{ij}(\hat{\boldsymbol{x}} \boxplus \Delta\tilde{\boldsymbol{x}}) \simeq \hat{e_{ij}} + \hat{J}_{ij}\Delta\tilde{\boldsymbol{x}} \tag{6}$$

where

$$\hat{J}_{ij} = \frac{\partial e_{ij}(\hat{\boldsymbol{x}} \boxplus \Delta\tilde{\boldsymbol{x}})}{\partial \Delta\tilde{\boldsymbol{x}}}$$

represents the Jacobian in the manifold. Thus, the local approximation of the objective function is of form

$$F(\boldsymbol{x} \boxplus \Delta\boldsymbol{x}) = \sum_{i,j \in C} F_{ij}(\hat{\boldsymbol{x}} \boxplus \Delta\tilde{\boldsymbol{x}}) = \sum_{i,j \in C} e_{ij}(\hat{\boldsymbol{x}} \boxplus \Delta\tilde{\boldsymbol{x}})^T \Omega_{ij} e_{ij}(\hat{\boldsymbol{x}} \boxplus \Delta\tilde{\boldsymbol{x}}),$$

which according to equation (6) can be expressed as follows:

$$\begin{aligned} F(\boldsymbol{x} \boxplus \Delta\boldsymbol{x}) &\simeq \sum_{i,j \in C} (\hat{e_{ij}} + \hat{J}_{ij}\Delta\tilde{\boldsymbol{x}})^T \Omega_{ij} (\hat{e_{ij}} + \hat{J}_{i_j}\Delta\tilde{\boldsymbol{x}}) \\ &\simeq \sum_{i,j \in C} \underbrace{\hat{e_{i_j}}^T \Omega_{ij} \hat{e_{i_j}}}_{\mathrm{c}_{ij}} + 2 \underbrace{\hat{e_{i_j}}^T \Omega_{ij} \hat{J}_{i_j}}_{\boldsymbol{b}_{ij}} \Delta\tilde{\boldsymbol{x}} + \Delta\tilde{\boldsymbol{x}}^T \underbrace{\hat{J}_{i_j}^T \Omega_{ij} \hat{J}_{i_j}}_{\boldsymbol{H}_{ij}} \Delta\tilde{\boldsymbol{x}} \end{aligned}$$

Removing the sum and rearranging leads us to a following quadratic equation:

$$F(\boldsymbol{x} \boxplus \Delta\boldsymbol{x}) \simeq c + 2\boldsymbol{b}^T \Delta\tilde{\boldsymbol{x}} + \Delta\tilde{\boldsymbol{x}}^T \boldsymbol{H} \Delta\tilde{\boldsymbol{x}},$$

which can be minimized by normal Gauss-Newton approach by solving the following linear system. This minimization yields the increment direction for a single iteration of the Gauss-Newton from

$$\boldsymbol{H}\Delta\tilde{\boldsymbol{x}}^* = -\boldsymbol{b}$$

by solving for $\Delta\tilde{\boldsymbol{x}}^*$. The single iteration solution $\boldsymbol{x}^*$ is thus obtained by re-mapping the increment $\Delta\tilde{\boldsymbol{x}}^*$ to original non-euclidean space with $\boxplus$. Instead of adding the increment normally to the current estimate, we obtain solution as:

$$\boldsymbol{x}^* = \boldsymbol{x} \boxplus \Delta\tilde{\boldsymbol{x}}^*$$

using 'boxplus' operator instead of normal summation.

(Grisetti et al., 2010; Hertzberg et al., 2013)

4 Results

The test data was collected from Evo forest study area located in Hämeenlinna, Finland. The overall study area contains roughly 120 sample plots of size 32x32 meters Boreal Forest Zone environment. The main tree species appearing in Boreal Forest Zone environment are pine, spruce and birch.

In this work 5 different 32x32 meters plots were analyzed and SLAM-correction was applied for every plot independently. In Table 2 the key attributes for each plot and reference data information is presented.

Table 2: Measurement information for each forest plot located in test site Evo

Plot number	1	2	3	4	5
Trajectory length (m)	380.3	415.2	527.3	420.5	326.1
Trajectory duration (s)	345	399	633	448	351
Number of trees	95	197	281	136	75
Number of observations	225	263	226	182	474
Initial plot height error (m)	0.528	2.955	2.866	1.310	0.645

The input data for SLAM-framework was collected AkhkaR4DW VUX-1HA laser scanner together with NovAtel Pwrpak7 GNSS-IMU localization equipment mounted on a backpack. The measurement frequency for point cloud data and trajectory data was 200 Hz. The point density for the laser scanner was set to 5085 points for single measurement with the scanning angle of 360 degrees with scanning surface roughly perpendicular to the moving direction. The average moving speed for the of the backpack scanner is roughly 1 m/s in the forest terrain. The whole backpack-mounted equipment for data collection is shown in Picture 2.

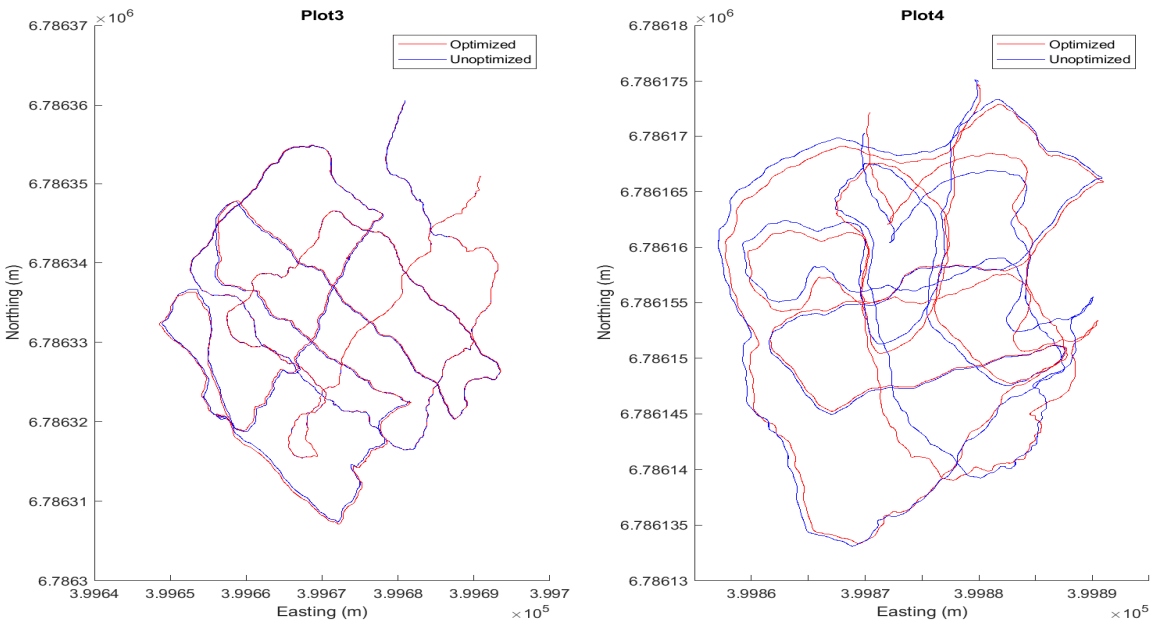

Figure 6: Plot 3 and Plot 4 unoptimized and optimized trajectory aligned from top view

Computational tools used for Forest Graph-SLAM included Matlab-script for collecting tree observations as well as Python-script that creates the graph from observations and original trajectory and sends it trough the g2o-optimizer. The observations are obtained by performing tree detection process in consecutive 10 second time windows of a single plot point cloud to create duplicate observations from single tree. The tree observations input for graph creation and trajectories

are provied all in an easily accessible text file format. Point cloud data, on the other hand, is in a .laz-file format that was accessed by a separate C++-script from Matlab-environment. All the visualization results are created using external software TerraScan and Matlab-visualization tools. Examples of aligned unoptimized and optimized graph are shown in Figure 6 and in Appendix B.

The automated algorithm for collecting tree locations in 10 second time windows of the original point cloud uses arc detection method developed by Eric Hyyppä. (Hyyppä et al., 2020) The parameters for the tree detection were set so that only observations with over 0.5m radius were considered, 90% of an inlier ratio for circle fitting was required as well as 30% of the arc length. This improves the reliability of the tree detection but decreases the amount of trees detected. To hasten the detection process, the point cloud data was filtered so that only points with 3-3.5 meters from detected ground were considered. The ground level detection for the filtering was performed using external software TerraScan based on point cloud triangulation. (TerraSolid, 2021)

Table 3: Initial internal point cloud accuracy and Forest Graph Slam -optimized internal point cloud accuracy for each forest plot. Here the errors are calculated as a distance (m) from tree observation in a cluster to the center of the cluster. For error statistics, all of the tree observations from all clusters were taken into account.

Plot number	1	2	3	4	5
Mean error (initial)	0.353	0.687	0.361	0.759	0.645
Mean error (optimized)	0.005	0.002	0.002	0.006	0.013
Error STD (initial)	0.236	0.484	0.367	0.534	0.438
Error STD (optimized)	0.005	0.002	0.003	0.005	0.014
Max error (Initial)	1.117	2.070	1.180	2.025	2.292
Max error (optimized)	0.040	0.010	0.013	0.027	0.124

In order to measure the improvement of the point cloud accuracy, the internal accuracy of the Forest Graph Slam method was considered. In this work, the internal accuracy of a landmark (in our case, a tree) is the average planar distance from all clustered landmark observations presumably belonging to the same actual landmark to the center of that landmark cluster. The tree observations were clustered together by combining observations with similar radius, planar location and long enough time windows. The point cloud internal accuracy is then defined by averaging over all the tree observations and their respective clusters as in Table 3. Because the optimization process itself is not fixed into a specific orientation or location, comparing internal accuracy is more applicable than absolute accuracy comparison without up to date reference data. An example of comparison between original point cloud and optimized point cloud for single plot is shown in Picture 7. The overall results and statistics for the internal accuracy for each plot are shown in Table 3.

Additionally, the overall point cloud accuracy without landmarks was tested briefly using height error measurements at the planar crossings of scanner trajectory. The height error for a crossing is defined as a difference between the z-coordinates

(elevation) of the trajectories in planar crossings where by assumption their elevation should roughly be the same. The results of overall accuracy for each plot are shown in Table 4

Table 4: The overall point cloud accuracy (elevation component) and Forest Graph Slam -optimized overall point cloud accuracy (elevation component) for each forest plot. Here the errors are calculated as a difference between the z-coordinates of the trajectories in planar crossings. For error statistics, all of the planar trajectory crossings are taken into account.

Plot number	1	2	3	4	5
Mean error (initial)	0.528	2.955	2.866	1.310	0.645
Mean error (optimized)	0.373	0.789	2.015	0.690	0.085
Error STD (initial)	0.402	1.790	2.037	0.920	0.313
Error STD (optimized)	0.294	0.623	1.994	0.705	0.054
Max error (Initial)	1.298	5.902	5.372	2.907	1.325
Max error (optimized)	0.880	2.215	5.235	1.742	0.139

5 Summary

Using Forest Graph Slam framework, we were able to increase the accuracy of 5 point cloud data sets collected from 5 different forest test sites with backpack laser scanning equipment. To optimize the point cloud accuracy, we first created a graph representation of the data from original trajectory and tree observations from the original point cloud. Using graph-optimization, we obtained the optimized trajectory, which was then used to create optimized point cloud from the environment.

We confirmed the results by comparing the internal accuracy of the point clouds before and after the optimization process. The results show that the planar accuracy of tree observations was increased from decimeter level to millimeter level as presented in Table 2. We also confirmed the increased accuracy by plotting the point cloud using external software (Picture 7) to see that generally the observable tree trunks have indeed better accuracy after the process. The millimeter level accuracy for planar tree locations is sufficient to be used accurately for forest management purposes. Extracting individual tree attributes would in addition require accurate overall point cloud accuracy around the trees.

Although the internal accuracy of the tree observations was increased in all of the plots almost equally, the overall point cloud accuracy described with elevation error was lacking for some of the plots. We confirmed that smaller number of tree observations compared to actual trees prior the optimization results in worse overall accuracy of the point clouds (especially noticeable in Plot 3, in Figure 6). This is natural since less tree observations results in less constraints in the optimization process potentially decreasing the effect of optimization process. To overcome this, one could perform the arc detection for the original point cloud with softer constraints to create more tree observations. However, this can potentially result in more false

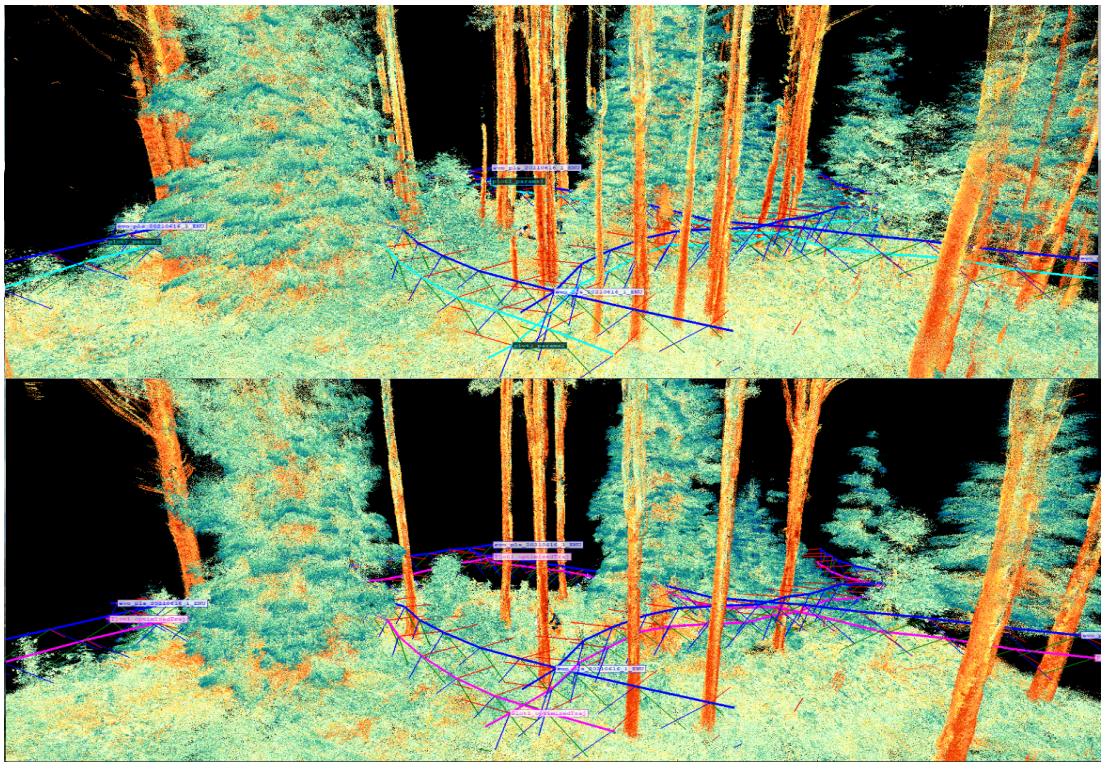

Figure 7: Example of accuracy improvement in part of Plot 1 point cloud visualization. The upper picture represents the unoptimized point cloud and the bottom picture represents the optimized point cloud. Dark blue plot represents the unoptimized trajectory. Pink and turquoise both represent the same optimized trajectory. This figure was produced by Antero Kukko

positive trees, which could create local anomalies in the optimized point cloud. The optimal parameters for finding the tree observations should then be confirmed with trial and error with respect to number of trees in the reference data.

Another potential improvement for the algorithm would be to use the Forest Graph Slam framework iteratively for a single point cloud multiple times. The first iterations of the framework would potentially fix the initial elevation errors in the point cloud so that the number of observed trees would increase in the further iterations. Since lack of tree observations in different time windows is mostly due to failures in ground level extraction, further improvements in the ground extraction algorithm would also be useful. As a whole, the arc detection is stable enough to provide reasonable tree observations if the elevation errors are small enough. Improving ground detection algorithm or using iterative framework is probably the most useful approach in future Forest Graph Slam development.

Extracting individual tree attributes for forest management purposes is possible if both the internal, and overall accuracy of the point cloud suffice. This was the case for most of the plots in our test site after the optimization. In the future with some improvements to the framework, Forest Graph Slam could provide a stable source of forest inventory data from backpack laser scanners in any forest environment.

References

[1] I. Balenović, X. Liang, L. Jurjević, J. Hyyppä, A. Seletković, and A. Kukko. Hand-held personal laser scanning – current status and perspectives for forest inventory application. *Croatian Journal of Forest Engineering*, 42(1):165–183, 2021.

[2] L. Chang, X. Niu, T. Liu, J. Tang, and C. Qian. Gnss/ins/lidar-slam integrated navigation system based on graph optimization. *MDPI Remote Sensing*, 1009 (11), 2019.

[3] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part 1. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[4] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[5] G. Grisetti, R. Kummerle, H. Strasdat, K. Konolige, and B. Wolfram. g2o: A general framework for graph optimization. 2011. Updated 9.5.2011. Cited 14.6.2021. Available: https://www.cct.lsu.edu/~kzhang/papers/g2o.pdf.

[6] G. Grisetti, R. Kummerle, H. Strasdat, K. Konolige, and B. Wolfram. g2o: A general framework for (hyper) graph optimization. 2017. Updated 11.3.2017. Cited 14.6.2021. Available: https://github.com/RainerKuemmerle/g2o/blob/master/doc/g2o.pdf.

[7] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 14(1):57–77, 2013.

[8] E. Hyyppä, A. Kukko, R. Kaijaluoto, J.C. White, M.A. Wulder, J. Pyörälä, X. Liang, X. Yu, Y. Wang, H. Kaartinen, J. Virtanen, and J. Hyyppä. Accurate derivation of stem curve and volume using backpack mobile laser scanning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 161(3):246–262, 2020.

[9] E. Hyyppä, X. Yu, H. Kaartinen, T. Hakala, A. Kukko, M. Vastaranta, and J. Hyyppä. Comparison of backpack, handheld, under-canopy uav, and above-canopy uav laser scanning for field reference data collection in boreal forests. *MDPI Remote Sensing*, 3327(12), 2020.

[10] J. Hyyppä, X. Yu, Harri. Kaartinen, A. Kukko, A. Jaakkola, X. Liang, Y. Wang, M. Holopainen, M. Vastaranta, and H. Hyyppä. *Topographic Laser Ranging and Scanning: Principles and Processing (2nd Edition)*. CRC Press, 2017.

[11] A. Jaakkola, J. Hyyppä, X. Yu, A. Kukko, H. Kaartinen, X. Liang, H. Hyyppä, and Y. Wang. Autonomous collection of forest field reference—the outlook and a first step with uav laser scanning. *MDPI Remote Sensing*, 785(9), 2017.

[12] L. Jelineck. *Graph-based SLAM on Normal Distributions Transform Occupancy Map*. Charles University - Faculty of Mathematics and Physics (Bachleor's Thesis), 2016.

[13] A. Kukko, R. Kaijaluoto, H. Kaartinen, V.V. Lehtola, A. Jaakkola, and J. Hyyppä. Graph slam correction for single scanner mls forest data under boreal forest canopy. *ISPRS Journal of Photogrammetry and Remote Sensing*, 132(10):199–209, 2017.

[14] A. Kumar. An introduction to simultaneous localization and mapping (slam) for robots. *Control Automation*, 2020.

[15] E. Næsset, T. Gobakken, J. Holmgren, H. Hyyppä, J. Hyyppä, M. Maltamo, M. Nilsson, H. Olsson, Å. Persson, and U. Söderman. Laser scanning of forest resources: The nordic experience. *Scandinavian Journal of Forest Research*, 19: 482–499, 2004.

[16] M. Pierzchala, G. Giguère, and R. Astrup. Mapping forests using an unmanned ground vehicle with 3d lidar and graph-slam. *Computers and Electronics in Agriculture*, 145(2):217–225, 2018.

[17] C. Stachniss, J. John, and S. Thrun. *Springer Handbook of Robotics (2nd Edition)*. Springer, Cham, 2016.

[18] TerraSolid. Terrascan user guide: Ground. 2021. Updated 26.7.2021. Cited 1.9.2021. Available: https://terrasolid.com/guides/tscan/crground.html.

[19] S. Thrun and M. Montemerlo. The graphslam algorithm with applications to large-scale mapping of urban structures. *MDPI Remote Sensing*, 25(5-6), 2006.

[20] M. Vermeer. *Geodesia: Kaiken perusta*. Aalto University - Department of Built Environment, 2019.

A Quaternions

Quaternions are an extension of complex numbers, usually represented as

$$a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$$

where $a, b, c, d \in \mathbb{R}$ and $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are the basic quaternions. Quaternions have similar properties as complex numbers with one important difference: the multiplication of quaternions is noncommutative. This can be seen from the fundamental formula for quaternion multiplication

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1 \tag{A1}$$

which shows the relationship between the basic quaternions and their similarity to complex numbers. All of the basic computation rules along with the multiplication can be derived from the equation (A1).

An alternative interpretation of the quaternions is a combination of a vector and scalar

$$\begin{bmatrix} s \\ \boldsymbol{v} \end{bmatrix}$$

where s is a scalar and $v \in \mathbb{R}^3$ This interpretation is especially useful with practical applications of quaternions such as representing rotations in 3-dimensional space. Technically, every rotation in $\mathbb{R}^3$ can be represented as rotation around some axis represented as unit vector. This interpretation requires 4 parameters, 3 for the unit vector in $\mathbb{R}^3$ and 1 for the rotation angle. The overall representation can be achieved by the use of unit quaternion

$$q = Q(\boldsymbol{u}, \theta) = \begin{bmatrix} cos(\theta/2) \\ \boldsymbol{u} sin(\theta/2) \end{bmatrix}$$

where $\boldsymbol{u}$ is the unit vector and θ is the rotation angle.

It should be noted that rotations can also be represented using only 3 parameters using Euler's angles that would provide easier computations. However, the use of Euler's angels can produce singularities in some processes. Using 4 parameters to represent rotations is therefore a more stable approach.

B Optimized trajectories

In this appendix the unoptimized and optimized trajectories (Pictures B1-B2) are presented from top view. Note that trajectories are only planar visualizations of pose variables' location components $t_i \in \mathbb{R}^3$.

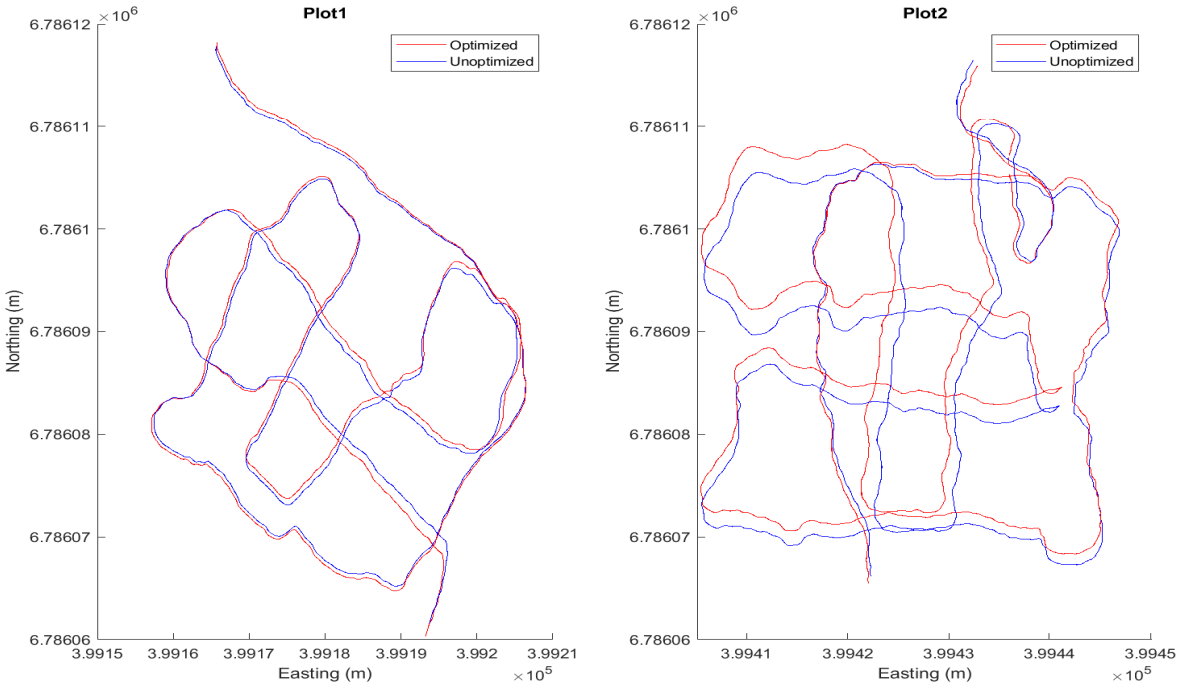

Figure B1: Plot 1 and Plot 2 unoptimized and optimized trajectory aligned

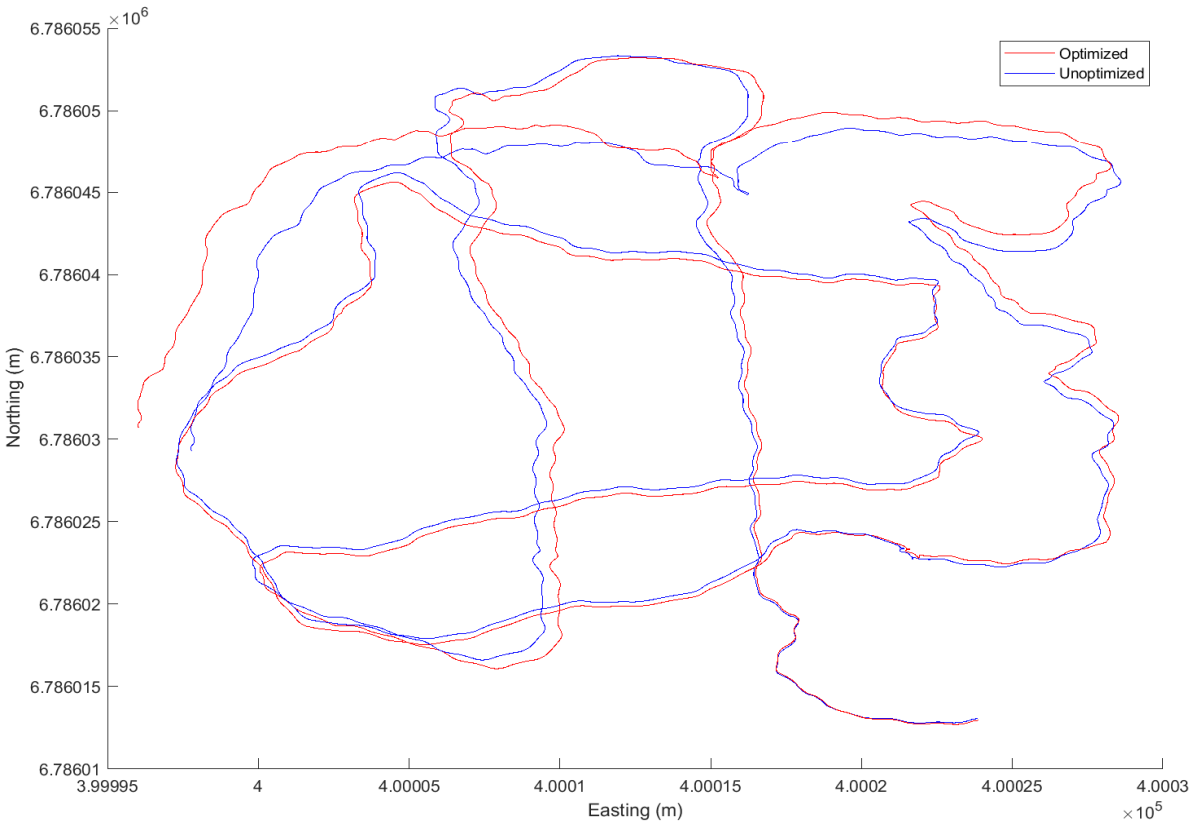

Figure B2: Plot 5 unoptimized and optimized trajectory aligned