DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2016

Integration of IMU and Velodyne LiDAR sensor in an ICP-SLAM framework

ERIK ZHANG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Integration of IMU and Velodyne LiDAR sensor in an ICP-SLAM framework

E R I K Z H A N G

Master's Thesis in Optimization and Systems Theory (30 ECTS credits)
Master Programme in Applied and Computational Mathematics
(120 credits)
Royal Institute of Technology year 2016
Supervisor at FOI: Fredrik Bissmarck
Supervisor at KTH: Xiaoming Hu
Examiner: Xiaoming Hu

TRITA-MAT-E 2016:66
ISRN-KTH/MAT/E--16/66--SE

Royal Institute of Technology
SCI School of Engineering Sciences

KTH SCI
SE-100 44 Stockholm, Sweden

URL: www.kth.se/sci

Abstract

Simultaneous localization and mapping (SLAM) of an unknown environment is a critical step for many autonomous processes. For this work, we propose a solution which does not rely on storing descriptors of the environment and performing descriptors filtering. Compared to most SLAM based methods this work with general sparse point clouds with the underlying generalized ICP (GICP) algorithm for point cloud registration. This thesis presents a modified GICP method and an investigation of how and if an IMU can assist the SLAM process by different methods of integrating the IMU measurements. All the data in this thesis have been sampled from a LiDAR scanner mounted on top of an UAV, a car or on a backpack. Suggested modification on GICP have shown to improve robustness in a forest environment. From urban measurements the result indicates that IMU contributes by reducing the overall angular drift, which in a long run is contributing most to the loop closure error.

Sammanfattning

Lokalisering och kartläggning (SLAM) i en okänd miljö är ett viktigt steg för många autonoma system. Den föreslagna lösningen är inte beroende på att hitta nyckelpunkter eller nyckelobjekt. Till skillnad från många andra SLAM baserade metoder så arbetar denna metod med glesa punktmoln där 'generalized ICP' (GICP)algoritmen används för punktmolns registrering. I denna uppsats så föreslås en variant av GICP och undersökner, ifall en tröghetssenspr (IMU) kan hjälpa till med SLAM-processen. LiDAR-data som har använts i denna uppsats har varit uppmätta från en Velodyne LiDAR monterat på en ryggsäck, en bil och på en UAV. Resultatet tyder på att IMU-data kan göra algoritmen robustare och från mätningar i stadsmiljö så visar det sig att IMU kan hjälpa till att minska vinkeldrift, vilket är det största felkällan för nogrannhet i det globala koordinat systemet.

Acknowledgements

I would like to thank my supervisor Fredrik Bissmarck for giving me the support needed and opportunity to write my master thesis about an interesting topic. Christina Grönwall for proofreading my thesis. Gustaf Hendeby for advices on the EKF. Jonas Nordlöf, Michael Tulldahl, Håkan Larsson for providing me with the data gathering and tools to process it. I would also like to thank my examiner Prof. Xiaoming Hu at KTH for making this thesis possible.

Contents

0.1 Abbreviations

SLAM ..... Simultaneous Localization and Mapping

ICP ..... Iterative closest point

GICP ..... generalized ICP

mGICP ..... Modified GICP

EKF ..... Extended Kalman Filter

LSO ..... Local sequence optimization (Section 2.4)

IMU ..... Inertial measurement unit

BFGS ..... Broyden–Fletcher–Goldfarb–Shanno

LiDAR ..... Light Detection And Ranging

DoF ..... degrees of freedom

UAV ..... Unmanned aerial vehicle

Chapter 1

Introduction

Simultaneous localization and mapping is a process which determines one's own location in an unknown area while at the same time generating a map of the area without access to external position systems or external reference markers. This process is mainly divided into two types, a full SLAM where one determines the map based on all available data such that order of measuring does not matter. The second type is the online SLAM which is used in this thesis, where the map is continuously built on top of as more measurement is available.

The input data type can come from any camera source but the data worked on here will be based on a Velodyne LiDAR (Light Detection And Ranging) which is a time of flight camera. From the inertial-measurement unit (IMU), observations from the accelerometer and gyroscope are used.

SLAM methods based on LiDAR instead of using photogrammetry are of interest as LiDAR measurement are less constrained when it is possible to collect data. LiDAR data can be collected at any time, day and night while low ambient light or shadows can hinder the photogrammetry process. Other benefits with LiDAR are that the LiDAR scanner can penetrate vegetation and has a higher range accuracy.

A 6DoF motion model is constructed to perform the state estimation where the IMU data and the processed LiDAR data are used to update the state estimation via an extended Kalman filter. The method used to process LiDAR data will be based on point cloud to a point cloud comparison where the underlying optimization cost function is based on the generalized iterative closest point (GICP) algorithm. The algorithm works with general point clouds and as such it does not rely on finding features such as corners, edges, or objects.

1.1 Purpose

The objective of this thesis has been to propose a SLAM framework which can generate an accurate 3D map where sparse point clouds are registered. The sparse point clouds are sampled by Velodyne Lidar scanners (see below Section 3.1.1). Then investigate if the proposed method can be improved by utilizing IMU measurement and how to integrate IMU measurement into the SLAM framework. This process could then in the future be used to generate a preliminary 3D map of a measurement where environmental details are unknown.

1.1.1 Method

The suggested algorithm is to set up an EKF model as a basis for our pose estimation, and then utilize IMU measurement and processed LiDAR data to update the pose estimation. The LiDAR data are processed by performing point cloud registration were the results indicates the relative position of two measurements, the evaluation of how well a certain alignment is will be based on the generalized ICP method.

To investigate how IMU could assist in the SLAM process, we use it to update the motion model for a better prediction. This may lead to cloud alignment ending in a more correct local minima solution. For the cost function it could help by improving the estimation of the uncertainty in our predicted pose which can be used directly in the cost function.

A pose graph is made were the vertices are the poses and the edges the relative position of the pose. A graph optimization is then proposed which utilize data which are driftless in time, where the error is distributed by alternating the active edges in the pose graph. The accelerometer can estimate roll and pitch with only a small bias from cloud registration, and a demonstration is also made by distributing the measured local error from loop closure.

Primarily only two data sets were used to tune in important parameters in the proposed SLAM method, but it should still represent different conditions as the data comes from different environments. One data set is from measuring a forest with a Velodyne VLP 16, and the other set is an urban environment with twice more details from a Velodyne HDL 32.

Limitations From the IMU only accelerometer and gyroscope were used to evaluate how internal sensors can contribute and magnetometer or barometer were not used. For each measurement only one type of sensor was attached so comparison if a higher quality sensors would help were not evaluated. The proposed SLAM method is not compared to other SLAM algorithm based on the same data set, as this thesis were directed towards investigating how and if IMU data can be used to improve the process.

1.2 Related work

Work related to registration and SLAM are presented here together with our own proposed framework.

1.2.1 SLAM process

Most SLAM process depends on finding certain type of features and then storing them in an EKF where the filter keeps track of possible location for each feature. This is where the proposed method mainly differs, by not storing any feature in the EKF. In common with most SLAM methods are that a source and a target point cloud is extracted and then some form of ICP algorithm is applied from which the result is used to update the current state estimation.

A common strategy to match point set is to apply a feature detection, such as finding edges, corners, or making some object classification and store them in the EKF [12], [17], [18], [37]. Other features are lines or curves [3], [21], [31] or converting them to points

[13]. SLAM has been attempted in difficult environments such as under water with sonar [9],[29], [29], vision denied environments [16], [18] and by radar [24]. There are work on SLAM using UAV platform [2], [6], [28], but those exploit a priori information of the environment. Alternatives to EKF have also been tested such as the information filter (IF) or the sparse extended information filter (SEIF) [35], [3], [7].

Most previous work have in common that they work with relative high resolution in both vertical and horizontal directions, and as number of features increases the complexity has quadratic growth [22]. Undetected features can be removed to reduce the complexity growth, but then it loses the ability to loop close as it cannot differentiate between a new and a removed feature. So a potential advantage by not relying on features are avoiding storing and finding features which lowers the processing cost in the EKF. Other advantage is being able to work with general point clouds and not being limited to an environment. Disadvantage includes possible increased cost of the point cloud registration process as the whole point cloud is used.

Featureless approaches which is similar to the proposed method have been proposed by Nuchter et al. in [1], [27], [26]. They worked on the SLAM problem using GPS and odometer. They used a stop and go approach for taking measurement and used a LiDAR which yields dense point clouds, thus they have found the ICP algorithm being good enough for the point cloud registration. Various methods for downsampling the cloud are applied and different approaches to store the cloud for a faster nearest neighbor search is tested. In [14], [36], [39], the ICP algorithm is also used for point cloud registration. A 3D map is generated with limited point density for which the measured point cloud is matched against. The latter two create a new local 3D map once the size is large enough and when the measurement is done the local maps is registered towards each other. Work on less dense point clouds have been done in [25]. The map is stored in a 3D grid system with a corresponding normal estimation for each cell, where a point to surface approach is used for the point cloud registration.

1.2.2 Point cloud registration

Once the point cloud of either features or points have been filtered out from a measurement, they are put into a source point cloud. A target point cloud is then made based on features from the EKF or based on points from previous measurement. An algorithm is then applied aligning the source point cloud relative to the target point cloud. The most common method is to use the iterative closest point method in the nominal paper by Besl and McKay in 1992 [4]. The method associates point pairs in source and target point clouds, and for features it would associate a measured feature to a feature in the EKF. For a given association it returns the best possible transform. This is usually good enough for feature based SLAM as feature points are usually sparse and there is less noise so given a decent prediction the association are usually correct.

Problems with ICP when working with point clouds, is the pair matching process, how to associate a point in a source point cloud to a target point cloud. As the pair association depends on how the point clouds are aligned, the problem becomes a non-convex optimization problem with one local minima for each pair association. To improve the process, filters are usually applied to remove outliers such only features as edges or corners remains. A potential solution to the non-convex problem has been proposed in [40] with the global optimal ICP (go-ICP) which finds the global optimal value. It uses

a branch and bound method with predictions of lower and upper bounds for each subset.

For point clouds with uneven density (due to the geometry of a scanner) the ICP algorithm is usually unstable as it would prioritize moving dense area towards dense area. This can possible be solved by using an alternative cost function. Instead of minimizing the point to point distance, point to line is minimized [8] or point to plane [23], [25].

The base registration which we will be working with is the generalized ICP proposed in [32] which is a type of surface to surface registration process. A covariance matrix is used to represents the surface and as such the optimization process minimizes the Mahalanobis distance between two point clouds. An improvement of the version is done in [15] mainly in how the surface is approximated.

1.3 Outline

The following five chapters begin with a short introduction of its content. Chapter 2 describes the theory used and introduces different ways to use IMU data. Chapter 3 contains how the algorithm is implemented and the order of execution. Chapter 4 presents the result and in Chapter 5 a discussion of the result is done and suggestions of what could be done in future. At the last Chapter 6 has a summary of this thesis with conclusions.

Chapter 2

Theory

This chapter begins with the basis of point cloud registration and then follows up with the Kalman filter used to estimate the state and the prediction uncertainty. Section 2.2 describes the process of interpreting the result of cloud registration and integrate it with the Kalman filter. Section 2.3 continues with proposals to alter the registration method and then the chapter ends with a method to optimize a pose graph based on IMU measurement and loop closure error.

2.1 Point Cloud Registration

2.1.1 Iterative Closest Point

The iterative Closest Point algorithm [4] finds the best rotation and translation vector such that the difference between a set of point pairs are minimized. The first problem is determining which points are a pair, and the simplest is by a point to point metric, where a pair is simply the point in A which is closest to a point in B assuming A is the 'reference system' and B is the measured data.

Let the set $\{p_i\}_0^N \in P$ be the set of points in the target cloud and $\{q_i\}_0^M \in Q$ be the set of source points. Let L be the set of indices i such that $||p_{j_*i} - q_i|| < \delta$ holds, where j_*i is chosen such that

$$j_*i = \underset{j \in 1,\dots,N}{\operatorname{argmin}} ||q_i - p_j||_2, \tag{2.1}$$

(closest point in p compared with q_i) and δ maximum distance between two points. Then the optimization problem is to minimize the objective function

$$E(\boldsymbol{R}, \boldsymbol{t})) = \sum_{i \in L} e_i(\boldsymbol{R}, \boldsymbol{t})^2 = \sum_{i \in L} ||p_{j_*i} - (\boldsymbol{R}q_i + \boldsymbol{t})||^2, \tag{2.2}$$

where $\boldsymbol{R}$ is an orthogonal rotational matrix and $\boldsymbol{t}$ is a translation vector.

Let p' be the center of mass of the points p_L and q' the center of mass q_L, then the translation vector becomes $\boldsymbol{t} = p' - \underline{\mathrm{R}}q'$ as a function of the rotation matrix. If we let $p_{cL} = p_L - p'$ and $q_{cL} = q_L - q'$[1] then we can rewrite Equation (2.2) to

$$\underset{\boldsymbol{R}}{\operatorname{argmin}} \sum_{i \in L} ||p'_{cj_*i} - \boldsymbol{R}q_{ci}||^2, \tag{2.3}$$

[1] where L is still the set of indices i in Q which has a corresponding pair.

where the translations part can be handled later because we find the optimal rotation under the assumption that both centroids are located at the same position. Expanding the norm, we get

$$\underset{\boldsymbol{R}}{\operatorname{argmin}} \sum_{i \in L} p_{cj_*i}^T p_{cj_*i} + q_{ci}^T q_{ci} - 2 p_{cj_*i}^T \boldsymbol{R} q_{ci} = \underset{\boldsymbol{R}}{\operatorname{argmin}} \sum_{i \in L} -2 p_{cj_*i}^T \boldsymbol{R} q_{ci} \tag{2.4}$$

$$= \underset{\boldsymbol{R}}{\operatorname{argmin}} -2\mathrm{Tr}\left(\boldsymbol{R} \sum_{i \in L} p_{cj_*i} q_{ci}^T\right). \tag{2.5}$$

By using singular value decomposition (SVD) [2] we can maximize the trace by setting

$$N = \sum_{i \in L} p_{cj_*i} q_{ci}^T = U \Sigma V^T \Rightarrow R = V U^T, \tag{2.6}$$

where U and V are the left-singular and right-singular vectors of N.

2.1.2 Generalized ICP

The generalized ICP described here is the one found in [32], which makes a probabilistic model of how a point is measured. It keeps the same Euclidean distance for making pairwise associations. The algorithm is divided mainly into two parts, the first part estimates the covariance and the second part performs the computation to find the optimal rotation and translation.

To derive the generalized ICP cost function we begin with assuming that when we measure a point, we are taking a random point on a surface. The set we will compare is a measured source set which is matched against a reference target set. The source set [3] $\hat{P}$ is made up by points $\{p_i\}$ and we let for each point p_i in $\hat{P}$ inherit the normal distribution $p_i \sim \mathcal{N}(\hat{p}_i, C_i^P)$. The expected value $\hat{p}_i$ is the measured coordinate point and the covariance C_i^P is a covariance matrix which represents an approximately surface of the point i.

Surface

Let $p_i \sim \mathcal{N}(\hat{p}_i, C_i^P), q_i \sim \mathcal{N}(\hat{q}_i, C_i^Q)$ and C_i being the covariance matrix of a point, and then assume that the correct transformation is given by T^*, such that $\hat{p}_i = T^* \hat{q}_i$ with $d_i^{(\boldsymbol{T})} = p_i - \boldsymbol{T}^* q_i$. The error distribution when transforming with T^* is then

$$\mathrm{E}\left[d_i^{(\boldsymbol{T}^*)}\right] = \hat{p}_i - \boldsymbol{T}^* \hat{q}_i = 0, \tag{2.7}$$

$$\mathrm{Var}\left[d_i^{(\boldsymbol{T}^*)}\right] = \mathrm{Var}[p_i] - \mathrm{Var}[\boldsymbol{T}^* q_i] = C_i^P + \boldsymbol{T}^* C_i^Q \boldsymbol{T}^{*T}, \tag{2.8}$$

$$d_i^{(\boldsymbol{T}^*)} = \mathcal{N}(0, C_i^P + \boldsymbol{T}^* C_i^Q \boldsymbol{T}^{*T}), \tag{2.9}$$

and maximum likelihood estimator $\boldsymbol{T}$ is computed by

$$\boldsymbol{T} = \underset{\boldsymbol{T}}{\operatorname{argmax}} \sum_{i \in L} \log\left(p(d_i^{(\boldsymbol{T})})\right) = \underset{\boldsymbol{T}}{\operatorname{argmin}} \sum_{i \in L} d_i^{(\boldsymbol{T})^T} (C_i^P + \boldsymbol{T}^* C_i^Q \boldsymbol{T}^{*T})^{-1} d_i^{(\boldsymbol{T})}. \tag{2.10}$$

[2] (Multiple method of solving the optimization problem exist where Eggert et al. [10] have compared and no method stands out so the SVD is used.)

[3] P and Q in this context is not related to the EKF.

The covariance of the theoretical sample methods represents a 3 dimensional Gaussian distributed probability where each axis represents the confidence in each direction. In the nominal paper [32] the covariance matrix is

$$C_i = R_i \begin{bmatrix} 1,0,0 \\ 0,1,0 \\ 0,0,\epsilon \end{bmatrix} R_i^T, \tag{2.11}$$

where R_i is the rotation matrix which projects the normal vector of a point i onto the third row. $\epsilon gives the relative uncertainty of the normal direction$ where a low value represents a high confidence in the normal direction. If the approximated normal of a point i is given by $\vec{n_i}$, then the matrix R_i is determined by

$$R_i = \begin{bmatrix} \vec{r_1} \\ \vec{r_2} \\ \vec{r_3} \end{bmatrix}, \tag{2.12}$$

$$\vec{r_3} = \vec{n_i}, \tag{2.13}$$

$$\vec{r_2} = \frac{\vec{e_y} - (\vec{e_y} \cdot \vec{n_i})\vec{n_i}}{||\vec{e_y} - (\vec{e_y} \cdot \vec{n_i})\vec{n_i}||}, \tag{2.14}$$

$$\vec{r_1} = \vec{r_3} \times \vec{r_2}, \tag{2.15}$$

$\vec{e_y}$ a unit normal in y direction $(0,\ 1,\ 0)$[4]. In [32] the method of determining the normals $\vec{n_i}$ are by performing a PCA (Principal component analysis)[11] analysis on the 20 nearest neighbors, and letting the eigenvector which corresponds to the smallest eigenvalue be the approximated normal vector. Due to the sparsity density of points one could get bad estimates if it attempts to find a plane based on a curve as seen in Figure 2.1.

An alternative way of approximating the normals have been done in [15] is to utilize the information that the manifold of the 3D environment is projected on the rotating 2D scanner as in Figure 2.2. Based on the manifold, a quad mesh is made and then letting the normal of a point be the average normal of surrounding meshes normal. This should result in a better approximation of the point normals as it is more representing the surroundings. But the drawback is that the normal approximation can only be based on its scan and not on previously recorded scans. Performance-wise this should also be faster if one knows the topology, by avoiding a nearest neighbor search as the topology is known within the scan.

Optimization function

To solve this optimization problem, the Equation (2.10) is solved by using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [5]. By defining

$$A = \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}, \tag{2.16}$$

the objective function can be rewritten as

$$V = \frac{1}{m} \sum_{i=0}^{m} \left(Ap_{s_i} - p_{t_i}\right)^T \left(C_T + RC_sR^T\right)_i^{-1} \left(Ap_{s_i} - p_{t_i}\right), \tag{2.17}$$

[4]The direction does not matter as the surface is symmetric.

Neighbours by nearest neighbor search

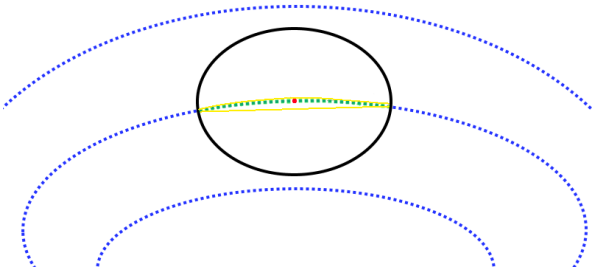


Figure 2.1: PCA algorithm searches for K nearest neighbors and if choosing to small values no neighbor from another scan lines is used to approximate the normal, which could yield bad normal approximation. Points used for normal approximation of the red point is in green, and blue are the LiDAR measurement.

Neighbours by 2D manifold neighbor search.

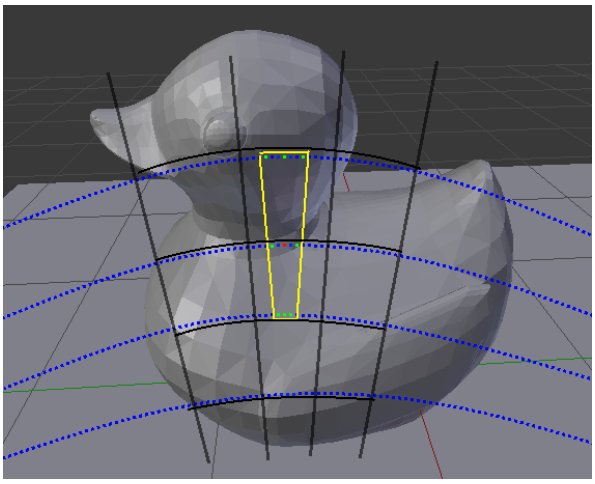


Figure 2.2: Normal approximation using the 2d manifold information, where the black lines represents the manifold. The chosen points to approximate the normal of the red points is in green color. Blue points are the LiDAR measurement.

Figure 2.3: Comparison on using PCA to approximate the normal vs using the 2d manifold information, the area spanned by the points is within the yellow border.

where m is number of corresponding pairs and index i goes through each pair. Rotation matrix R as defined in Appendix B and each point p is on the format $[x\ y\ z\ 1]^T$.

The derivative of V with respect to variable x is

$$\frac{\partial}{\partial x}\left(\frac{1}{m}\sum_{i=0}^{m}\left(Ap_{s_i}-p_{t_i}\right)^T\left(C_T+RC_sR^T\right)^{-1}\left(Ap_{s_i}-p_{t_i}\right)\right)= \left\{H_i=\left(C_T+RC_sR^T\right)_i^{-1}\right\}= \frac{1}{m}\sum_{i=0}^{m}\left(2\left(Ap_{s_i}-p_{t_i}\right)^T H_i\frac{\partial}{\partial x}\left(AP_{s_i}\right)+\left(Ap_{s_i}-p_t\right)^T\left(-H_i\right)\frac{\partial}{\partial x}(H_i)H_i\left(Ap_{s_i}-p_t\right)\right), \tag{2.18}$$

where we utilized H being a symmetric matrix. The derivative of Equation (2.18) with respect to a translation variable ($x \in T$) can be simplified to Equation (2.19).

$$\frac{\partial V}{\partial \vec{x}}=\left[\frac{\partial V}{\partial x}\quad\frac{\partial V}{\partial y}\quad\frac{\partial V}{\partial z}\right]=\left\{\frac{\partial Ap_{s_i}}{\partial \vec{x}}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{bmatrix}\right\}=\frac{2}{m}\sum\left(\left(Ap_{s_i}-p_{t_i}\right)^T H_i\right). \tag{2.19}$$

For differentiation with respect to the angle, the left part of plus sign in Equation (2.18) becomes

$$\frac{1}{m}\sum_{i=0}^{m}\left(2\left(Ap_{s_i}-p_{t_i}\right)^T H_i\frac{\partial}{\partial x}\left(AP_{s_i}\right)\right)=$$

$$\frac{1}{m}\sum_{i=0}^{m}\left(2\left(Ap_{s_i}-p_{t_i}\right)^T H_i\frac{\partial}{\partial \alpha}\left(AP_{s_i}\right)\right)=\left\{\frac{\partial}{\partial \alpha}\left(AP_{s_i}\right)=\begin{bmatrix}\frac{\partial R}{\partial \alpha} & 0\\ 0 & 0\end{bmatrix}p_{s_i}\right\}=$$

$$=\frac{2}{m}\sum_{i=0}^{m}\left(2\left(Ap_{s_i}-p_{t_i}\right)^T H_i\right)\begin{bmatrix}\frac{\partial R}{\partial \alpha} & 0\\ 0 & 0\end{bmatrix}p_{s_i}=\frac{\partial V}{\partial \vec{x}}\begin{bmatrix}\frac{\partial R}{\partial \alpha} & 0\\ 0 & 0\end{bmatrix}p_{s_i}, \tag{2.20}$$

where $\frac{\partial R}{\partial \alpha}$ solution is shown in appendix. As the function value is already determined, only one more matrix multiplication operation is required

For the right part of the equation, it involves solving

$$\frac{1}{m}\sum_{i=0}^{m}\left(\left(Ap_{s_i}-p_t\right)^T(-H)\frac{\partial}{\partial x}(H)H\left(Ap_{s_i}-p_t\right)\right), \tag{2.21}$$

$$\frac{\partial}{\partial \alpha}(H)=\frac{\partial}{\partial \alpha}(C_t+RC_sR^T)=\frac{\partial}{\partial \alpha}RC_sR^T, \tag{2.22}$$

but as each corresponding pair has an unique derivative it will involve a lot of additional operations. But assuming that the distance between Ap_{s_i} and p_{ti} is small as we always pair up points with the nearest points, and that we have a good initial guess, then we could approximate Equation (2.21) as zero.

2.2 The Kalman filter

The state estimation part is based on the extended Kalman filter [33] which is a non-linear version of the Kalman filter, which also divides the estimation into a prediction and a correction part. We are using the extended Kalman filter due to state coordinate system is global while the coordinate system of the IMU is local. The 6DoF model used has the state vector

$$\vec{x}=\begin{bmatrix}\vec{p}\\ \vec{v_p}\\ \vec{a_p}\\ \vec{\theta}\\ \vec{\omega}\end{bmatrix}, \tag{2.23}$$

where $\vec{p}$ is our absolute position in a 3D Cartesian space, $\vec{v}_p$ the velocity, $\vec{a}_p$ the acceleration, θ, angles (pitch, roll, yaw), and ω the angular velocity, all measured in extrinsic coordinates.

The state prediction equations for the extended Kalman filter are given as in [34]

$$\dot{\vec{x}}(t)=f\left(\vec{x}(t),u(t)\right), \tag{2.24}$$

$$\dot{P}(t)=FP+PF^T+Q, \tag{2.25}$$

where P is the covariance matrix, F is the Jacobian of f, $u(t)$ control input and Q the covariance matrix of model uncertainty. Corresponding correction equations are given as

$$\vec{y} = \vec{z} - \vec{h}(\vec{x}_{pri}(t)), \tag{2.26}$$

$$K = P_{pri}(t)H' \left(HP_{pri}(t)H^T + R\right)^{-1}, \tag{2.27}$$

$$\vec{x}_{post}(t) = \vec{x}_{pri}(t) + K\vec{y}, \tag{2.28}$$

$$P_{post}(t) = (I - KH)\, P_{pri}(t), \tag{2.29}$$

with $\vec{z}$ the measurement, $h(x(t)_{pri})$ the state in the measured coordinate system, H Jacobian of h. Subscript *pri* (priori) is the state before the correction is applied and *post* (posteriori) is the state after correction is applied.

For the prediction part, we assume that we have no input data and integrate the pose forward. Prediction equation (2.24) and (2.25) can be solved as

$$\dot{\vec{x}} = A\vec{x}, \tag{2.30}$$

$$\vec{x}(t + \Delta t) = e^{A \cdot \Delta t}\vec{x}(t), \tag{2.31}$$

$$P(t + \Delta t) = P + \Delta t(FP(t) + P(t)F^T + Q), \tag{2.32}$$

where we have assumed that we do not have the control input. The prediction covariance is solved by forward Euler with Δt as the time until the next IMU or LiDAR observation.

Since the states all share a common coordinate system, we have a very simple motion model as follows

$$\dot{\vec{p}} = I_{3\times3}\vec{v}_p, \tag{2.33}$$

$$\dot{\vec{v}}_p = I_{3\times3}\vec{a}_p, \tag{2.34}$$

$$\dot{\vec{\theta}} = I_{3\times3}\vec{\omega}, \tag{2.35}$$

which yields us the matrices in the prediction step as

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix}, \tag{2.36}$$

$$F = A, \tag{2.37}$$

$$Q = \begin{bmatrix} \frac{\Delta t^2}{2} I_{3\times3}\epsilon_{acc} & 0_{3\times3} & \cdots & \cdots & \cdots \\ 0_{3\times3} & \Delta t I_{3\times3}\epsilon_{acc} & 0_{3\times3} & \cdots & \cdots \\ \cdots & 0_{3\times3} & \epsilon_{acc} I_{3\times3} & 0_{3\times3} & \cdots \\ \cdots & \cdots & 0_{3\times3} & \Delta t \epsilon_{angvel} I_{3\times3} & 0_{3\times3} \\ \cdots & \cdots & \cdots & 0_{3\times3} & \epsilon_{angvel} I_{3\times3} \end{bmatrix}. \tag{2.38}$$

Matrix F happens to be equal to A as $\dot{x}$ is linearly dependent on x_{state_i}. Prediction error of acceleration is given by ϵ_{acc} and the prediction error of angular velocity is given by $\epsilon_{angvvel}$.

2.2.1 Correction by IMU observation

The correction equations as stated before

$$\vec{y} = \vec{z} - \vec{h}(\vec{x}_{pri}(t)), \tag{2.39}$$

$$K = P_{pri}(t)H' \left(HP_{pri}(t)H^T + R\right)^{-1}, \tag{2.40}$$

$$\vec{x}_{post}(t) = \vec{x}_{pri}(t) + K\vec{y}, \tag{2.41}$$

$$P_{post}(t) = (I - KH)\, P_{pri}(t), \tag{2.42}$$

and in our case y is the difference between the measurement from IMU and the predicted values, K the Kalman gain and R the IMU noise. In Equation (2.39) the function h takes the current system state and rotates it to the current intrinsic system. The rotation matrices R_z, R_y, R_x can be found in the Appendix (B.1), and the state relation between intrinsic state s_i and extrinsic state s_e is given by

$$\begin{bmatrix} R_z R_y R_x \vec{e}_x & R_z R_y R_x \vec{e}_y & R_z R_y R_x \vec{e}_z \end{bmatrix} s_i = I_{3\times 3} s_e \Rightarrow s_i = (R_z R_y R_x)^T s_e, \tag{2.43}$$

with e_i being the unit vectors in extrinsic coordinate. Equation (2.39) becomes

$$\vec{z} - \vec{h}(x_{pri}) = \begin{bmatrix} m_{acc} - (R_z R_y R_x)^T \cdot \left(\hat{\vec{a}}_p + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right) \\ m_{angvel} - (R_z R_y R_x)^T \hat{\vec{\omega}} \end{bmatrix} \tag{2.44}$$

where m is the measured data from IMU where m_{acc} is the measured acceleration and m_{angvel} is the measured angular velocity in a local Cartesian coordinate system.

The H is the Jacobian of h and with $Rot = (R_z R_y R_x)^T$, each partial derivative is

$$\frac{\partial h_i}{\partial p_j} = 0, \tag{2.45}$$

$$\frac{\partial h_i}{\partial v_j} = 0, \tag{2.46}$$

$$\frac{\partial h_i}{\partial a_j} = \sum_{k \neq j} Rot_{i \bmod 4, k}(a_k + g\delta_{k3}) + Rot_{i \bmod 4, j}, \tag{2.47}$$

$$\frac{\partial h_i}{\partial \theta_j} = \sum_{k=1}^{3} \left(\frac{\partial Rot}{\partial \theta_j}_{\,i \bmod 4, k} a_k + \frac{\partial Rot}{\partial \theta_j}_{\,i \bmod 4, k} \omega_k \right), \tag{2.48}$$

$$\frac{\partial h_i}{\partial \omega} = \sum_{k \neq j} Rot_{i \bmod 4, k}\omega_k + Rot_{i \bmod 4, j}, \tag{2.49}$$

$$\forall\ i\ \in \{1, 6\} \tag{2.50}$$

$$\forall\ k\ \in \{1, 3\}, \tag{2.51}$$

$$\delta_{uy} = 0 \text{ if } u \neq y \text{ else } 1, \tag{2.52}$$

with R_{ij} being the element in row i and column j in rotation matrix Rot. $k = 1$ is relative the x axis, $k = 2$, the y axis and $k = 3$ the z axis. The derivatives of the rotation matrix is given in the Appendix B.

The measurement uncertainty matrix R has the form

$$R = \begin{bmatrix} I_{3\times3}\sigma_{acc} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3}\sigma_{angvel}, \end{bmatrix} \tag{2.53}$$

where σ_{acc} the variance in measurement error of accelerometer and σ_{angvel} the variance in measurement error of gyroscope.

2.2.2 Correction by point cloud registration

The result of a point cloud registration, consist of a value of how well the comparison is and the relative position of a source cloud and a target cloud. One method is to update the EKF by interpreting the result as an absolute position (like a simulated GPS data measurement), so when updating the EKF one updates the x-y-z states (referred to PFB or Position feedback). Another interpretation is to treat the result as a relative difference compared to last pose, for which in a continuous system corresponds to updating the velocity states (referred to VFB or Velocity feedback).

By updating the EKF via position feedback, we would also update the variance of the position states even though we are not getting more confident in the position states. This makes the confidence of our state being higher than what it is. But by updating the state as the camera pose we would absorb potential drift between the EKF state location and the pose location which can be seen in Figure 2.4:A.

For the velocity feedback we would preserve uncertainty in our global position which would better represent the true uncertainty as we cannot measure the global position. By updating the EKF with a delta in position we could potentially introduce a drift error. If the point cloud registration detects a certain delta in position, and feedbacks to the EKF, then the state of the EKF will be different for the state of last measure pose. This difference can then accumulate such that the next measurement, the predicted state is a state relative last state not relative last measured pose (Figure 2.4:B shows the effect).

As the ICP based method only yields a local minima, the uncertainty matrix is assumed to be constant. Both feedback methods use the Equations(2.39) to (2.42) to correct the state estimation.

Position feedback

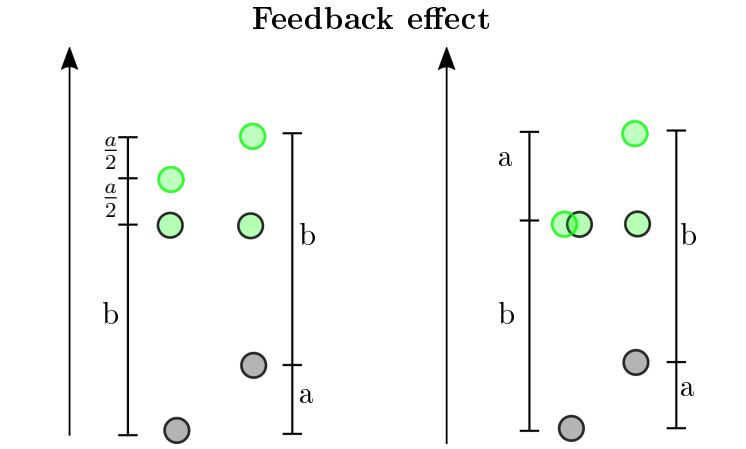
For position feedback the matrices for correction equation are set as follows

$$\vec{z} - \vec{h}(x_{pri}) = \begin{bmatrix} \vec{m}_{pos} - \vec{p} \\ \vec{m}_{ang} - \vec{\theta} \end{bmatrix}, \tag{2.54}$$

$$H = \begin{bmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \end{bmatrix}, \tag{2.55}$$

$$R = \begin{bmatrix} \tau_p^2 \cdot I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & \tau_a^2 \cdot I_{3\times3} \end{bmatrix}, \tag{2.56}$$

with m being the 'measured' position from the ICP algorithm in extrinsic system.



A. Position feedback. The drift between true pose and estimated pose is absorbed.

B. Velocity feedback. The drift between true pose and estimated pose remains.

Figure 2.4: Assuming the confidence of the prediction and observation is the same. Gray circle is the initial state, gray circle with filling is the predicted position of the following cloud. Green circle is the final position of the cloud. Left hand side is the state estimation and right side the 'true' pose. For our example, A. the drift differences halves, while for B. it is the same.

Velocity feedback

For velocity feedback the matrices for correction equation are set as follows

$$\vec{z} - \vec{h}(x_{pri}) = \begin{bmatrix} \vec{m}_{vel} - \vec{p} \\ \vec{m}_{ang} - \vec{\theta} \end{bmatrix}, \tag{2.57}$$

$$H = \begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \end{bmatrix}, \tag{2.58}$$

$$R = \begin{bmatrix} f \cdot \tau_p^2 \cdot I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & \tau_a^2 \cdot I_{3\times3}, \end{bmatrix} \tag{2.59}$$

where f is the frequency the LiDAR data are sampled.[5] The velocity from ICP measurement are determined by one step backward difference, which is the difference between previous and current pose multiplied by the scan frequency. As variance grows linearly with number of measurement the uncertainty in velocity is multiplied with the frequency.

2.3 Modified GICP

The cost function is altered in two different ways, first is to alter the covariance by adjusting how the surface is approximated. The second is to add additional cost to solutions far away from the initial guess, since we can estimate the confidence of the predicted pose.

[5] It is assumed that as GICP stops when some certain convergence criteria are met and therefore each measurement will have some error. So the accumulated variance per seconds becomes $f \cdot \tau_p^2$.

Surface

For our proposal on the surface approximation we assume that the confidence of the surface is inversely proportional to the size of the surface mesh. Instead of equation (2.11) we modified it to

$$C_i = R_i \begin{bmatrix} ||\vec{v}_{rl}||n_w/n_h & 0 & 0 \\ 0 & ||\vec{v}_{tb}|| & 0 \\ 0 & 0 & \epsilon \end{bmatrix} R_i^T m \tag{2.60}$$

where n_w is the number of points used to describe the surface width and n_h number of points used to describe the surface height. Vector $\vec{e}$ is the vector shown in Figure 2.5. This will allow greater mismatch along the $\vec{e}_{tb}$ direction relative the $\vec{e}_{rl}$ direction. This alternative surface approximation will be called the proportional surface approximation.

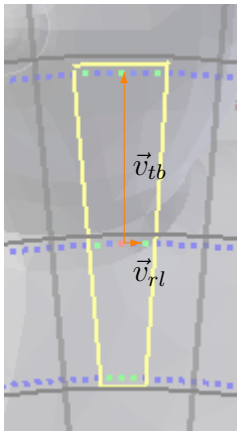


Figure 2.5: $\vec{e}_{rl}$ points to nearest point in 'right-left' direction and $\vec{e}_{tb}$ the nearest point in 'top-bottom' direction.

As before in equation (2.11) we determine the covariance with the same method, but as the surface is no longer symmetric the rotational matrix is adjusted and becomes

$$R_i = \begin{bmatrix} \vec{r_1} \\ \vec{r_2} \\ \vec{r_3} \end{bmatrix}, \tag{2.61}$$

$$\vec{r}_3 = \vec{n}_i \tag{2.62}$$

$$\vec{r}_2 = \frac{\vec{e}_{tb} - (\vec{e}_{tb} \cdot \vec{n}_i)\vec{n}_i}{||\vec{e}_{tb} - (\vec{e}_{tb} \cdot \vec{n}_i)\vec{n}_i||}, \tag{2.63}$$

$$\vec{r}_1 = \vec{r}_3 \times \vec{r}_2, \tag{2.64}$$

where we see that $\vec{r}_2$ is the $\vec{e}_{tb}$ projected on the surface spanned by the normal $\vec{n}_i$.

Cost function changes

The prediction step only gives a suggestion where to start the comparison, but no weight is put in on the confidence of the prediction. Even if the prediction would be perfect, the cloud registration algorithm would still find the local minima which could be located far

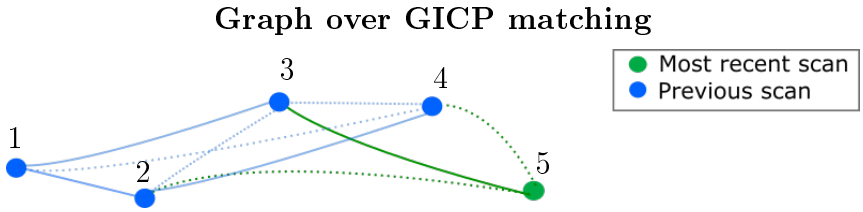


Figure 2.6: The vertices represents a pose and the measured relative pose is stored in the edges. Green color is for the latest scan and blue is the previous scan. Filled line is an active edge and a dotted ones are inactive edges.

from the correct location. To weight in the confidence of our prediction more, a penalty function based on the uncertainty is added such that (2.17) get replaces with

$$(1-\alpha)\frac{1}{m}\sum_{i=0}^{m}\left(Ap_{s_i}-p_{t_i}\right)^T\left(C_T+RC_sR^T\right)^{-1}\left(Ap_{s_i}-p_{t_i}\right)+(\alpha)x^TMx, \tag{2.65}$$

where x is the estimated transformation matrix A in vector form.[6] Variables α and the M matrix are used to weight appropriately between IMU-EKF prediction and GICP criteria. BFGS algorithm is applied as before, but the derivative of Equation (2.18) becomes

$$(1-\alpha)\frac{1}{m}\sum_{i=0}^{m}\left(2\left(Ap_{s_i}-p_{t_i}\right)^TH_i\frac{\partial}{\partial x}\left(AP_{s_i}\right)+\left(Ap_{s_i}-p_t\right)^T\left(-H_i\right)\frac{\partial}{\partial x}(H_i)H_i\left(Ap_{s_i}-p_t\right)\right) +\alpha\cdot 2x^TM. \tag{2.66}$$

2.4 Sequence optimization

Pairwise registration has an inherent flaw where a point cloud is only compared to one other point cloud at a time and not compared to a larger merged point clouds. If the compared target cloud has a pose error then the new position (pose) will be biased with the error. To address this problem a graph is created where every registration result is stored. Each vertices represent one pose and each edge represent one result from the cloud registration which stores the relative position. Per default the active connection used for each pose is the one which yielded the lowest cost value.

Then we could optimize the graph by choosing alternative links which yields an end state which has less cumulative error. This would allow more local error as long as in the *long run* being more correct. Just a method of spreading out errors, as long as the global error is smaller we accept more local errors. Information such as when we return to a known position as when closing a loop (more details Section 2.4), or using accelerometer and gyroscope to determine pitch and roll angles.

[6] Under assumption that during the initial alignment both the source and target cloud is translated such that source cloud pose center is at (0,0,0).

Local Sequence optimization (LSO)

If the edges stores the relative pose relative the compared frame, then the optimization objective becomes:

$$\min(E_d - C_d)^T T(E_d - C_d) \tag{2.67}$$
$$S_n = S_{c(n)} + R_e(S_{c(n)}) \times D(n, c(n)) \; \forall \; n \in 1, ..., \text{last} \tag{2.68}$$
$$C_d = S_{last}. \tag{2.69}$$

If the edges stores the relative pose relative previous frame, then the objective becomes

$$\min(E_d - C_d)^T T(E_d - C_d) \tag{2.70}$$
$$S_n = S_{n-1} + R_e(S_{n-1}) \times D(k(n)) \; \forall \; n \in 1, ..., \text{last} \tag{2.71}$$
$$C_d = S_{last}. \tag{2.72}$$

Where S_n is the pose of frame n, E_d is the targeted end state and C_d the current end state. T is matrix which weight the state variables towards each other. $c(n)$ is the choice which pose number n is compared against. R_e is the rotation matrix generated by the pose S_n and $D(n, c(n))$ is the matrix which holds the relative position between pose n and the pose $c(n)$. $D(k(n))$ is the relative position between pose n and $n-1$ given the sequence choice (k(n)).

Graph sequence optimization

For the graph optimization a slightly different cost function is used, where we wish to preserve key loop closure points by adding a penalty for each loop closure location instead of only the last frame. This can be formulated as the following optimization problem if the edges are stored relative to the optimal frame

$$\min \sum_{k \in LC_{pair}} (C_k - E_k)^T T(C_k - E_k) \tag{2.73}$$
$$S_n = S_{c(n)} + H(S_{c(m)}) \times D(n, c(n)) \; \forall \; n = 1, ..., \text{last} \tag{2.74}$$
$$E_k = H^{-1}(S_{v(k)})(S_k - S_{v(k)}) \forall k \in LC_{pair}, \tag{2.75}$$

or as

$$\min \sum_{k \in LC_{pair}} (C_k - E_k)^T T(C_k - E_k) \tag{2.76}$$
$$S_n = S_{n-1} + H(S_{n-1}) \times D(k(n)) \; \forall \; n \in 1, ..., \text{last} \tag{2.77}$$
$$E_k = H^{-1}(S_{v(k)})(S_k - S_{v(k)}) \forall k \in LC_{pair}. \tag{2.78}$$

If the edges are stored relative the previous frame. The set LC_{pair} is the set of pairs which connect to each other when the loop closes. Vertex given by $v(k)$ is the vertex (frame) connected to vertex (frame) k. When a connection is made at frame id k the value of C_k becomes the difference between $C_k = R_e^{-1}(S_{v(k)})(S_k - S_{v(k)})$. So we allow $C_j - E_j$ grow if it benefits more at another loop close location.

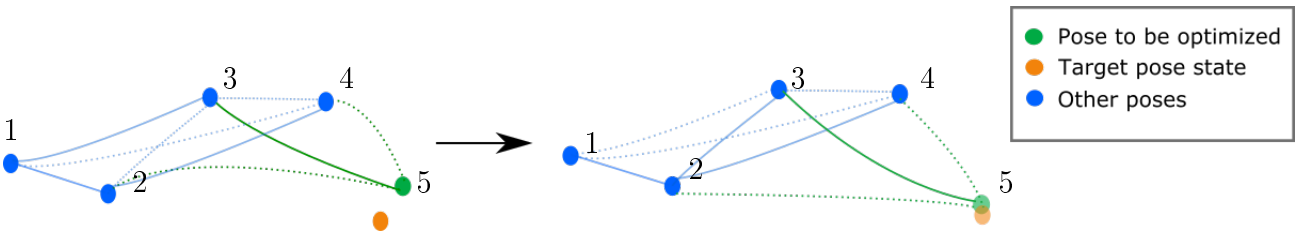


Figure 2.7: Vertices are poses and edges are the relative position. The green vertex is the pose to be optimize towards and the LSO algorithm swaps links and as long as the green vertex (to be optimized) converges to the target orange vertex. Blue is previous poses which can also swap links to improve the green vertex state.

2.4.1 Sequence solution method

For both graph sequential optimization and local sequential optimization there is a problem with multiple choice. The choices represent a non-convex integer programming problem, where the solution space grows with the power of number of vertices, so finding the *global optimal* solution is infeasible. But a method to find one local minima is to perturb each link in the graph, and as long as the cost function lowers down we keep the new perturbation which will yield a local optimal solution.

1. Determine the set of nodes which can affect the outcome
2. Perturb each choice and check the value function
3. If improved keep the change
4. Continue with next node

Chapter 3

Methods and Implementation

This chapter begins with the platform used to collect the data, and how the equipment was set up. The following section contains the evaluation method used to evaluate the result from ICP-SLAM algorithm. The third section is about the implementation of the ICP-SLAM algorithm with the parameter used. A summary of parameter are also available in Appendix A.

3.1 Experimental setup

3.1.1 3D Scanner

The LiDAR sensors used to collect data are Velodyne HDL 32 [38] and Velodyne VLP 16 [38]. Both sensors have 32 respective 16 time of flight lasers mounted in a column which have a vertical angular resolution of 4/3 degree and 2 degree respectively. As the column is mounted on top of a rotation platform each revolution will yield a 3D cloud with characteristical scan lines. Each rotation corresponds to one LiDAR measurement or one frame.

3.1.2 Forest (Backpack)

For the forest environment evaluation the Velodyne VLP 16 was mounted onto a backpack, together with an Xsens MTi 10 IMU. The sampling rate for Velodyne were set to ten images per second, which recorded approximately 1100×16 sized point clouds per revolution. The setup can be seen in Figure 3.1 and the data were sampled for about three minutes of walking paced movement.

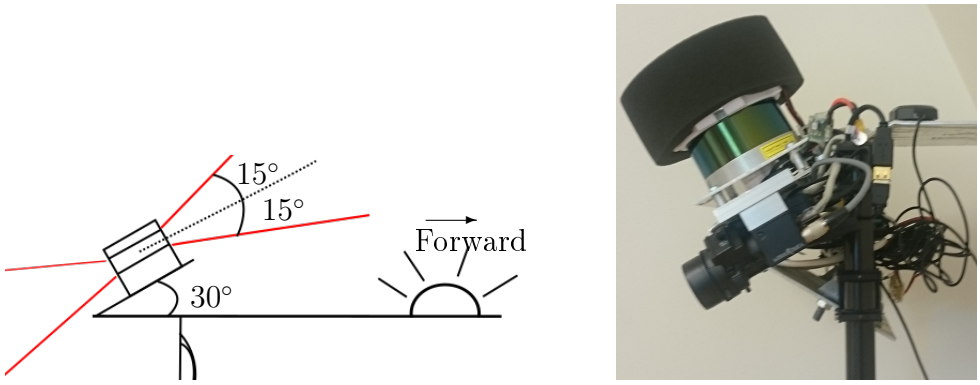


Figure 3.1: Setup for the forest measurement. Stereo camera used for the visual slam system.

3.1.3 Urban (Car)

The Velodyne HDL 32 were used and sampling rate set to 20 Hz which records approximately 610×32 points per revolution. The IMU used were the internal one made by three 2D INS. The value used were the average between those pointing the same direction. The setup can be seen in Figure 3.2. For the short loop the recording time for 2 laps were 4 minutes at a speed of 10-25 km/h. The longer loop were recorded for 14 minutes, with a speed of 30-50 km/h.

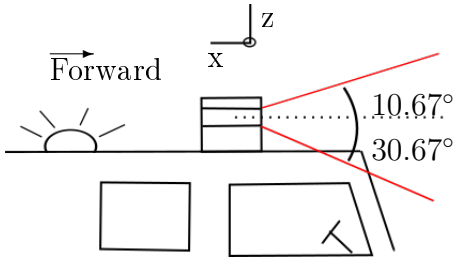


Figure 3.2: Setup for the urban measurement.

3.1.4 Countryside (Aerial)

For the aerial scenario we used a Velodyne VLP 16 mounted on a hexacopter with a sampling rate of 10 Hz, for IMU a Pixhawk were used. The setup can be seen in Figure 3.3.

Figure 3.3: Setup for the aerial measurement. Black box in front of Velodyne contains the Pixhawk IMU.

3.2 Evaluation

For evaluation of the results we will primarily look into the loop closure error as a metric, the distance between the same point measured at two different times. An attempt on determining if the position error is due to positional drift error or angular drift error is attempted, where the error is given in drift per meter travelled (percentage error), and angular drift in radian per meter travelled.

The generated pose path is also compared to a reference pose if available and for robustness evaluation the pose path is examined to determine if it is locally consistent.

Drift Error

For the positional drift we will compare the distance between poses on a straight line with the expected travel distance based on a map if available. The percentage drift is then determined by formula $\xi_{\text{pos}} = \frac{\text{traveldist} - \text{expecteddist}}{\text{expecteddist}}$.

To estimate the drift caused by angular errors in cloud registration, we assume that any error in loop closure minus the error of positional drift is caused solely by the angular drift. The positional error is the length of the loop multiplied by $\xi_{mathrmpos}$. Then the error caused by drift $\xi_{\text{drifterror}} = \xi_{\text{offset}} - \xi_{\text{pos}} \cdot \Delta\xi_{\text{Dist}}$.

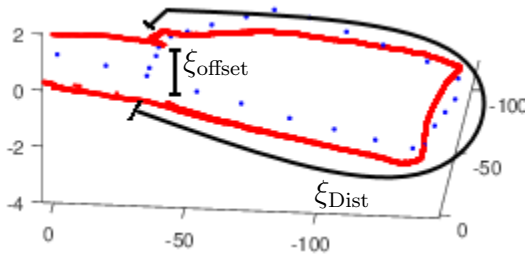


Figure 3.4: Example of loop closure error for the small urban data set. Blue dot is the reference position, red is the estimated trajectory.

The travel path is projected as if we were only travelling on the 2D plane and then we assume that the true travel path only travels along one direction. Then if we assume there is a constant angular drift per travelled distance with the initial error angle being zero then the path generated due to constant angular drift can be described as a circle in 2D plane as in Figure 3.5.

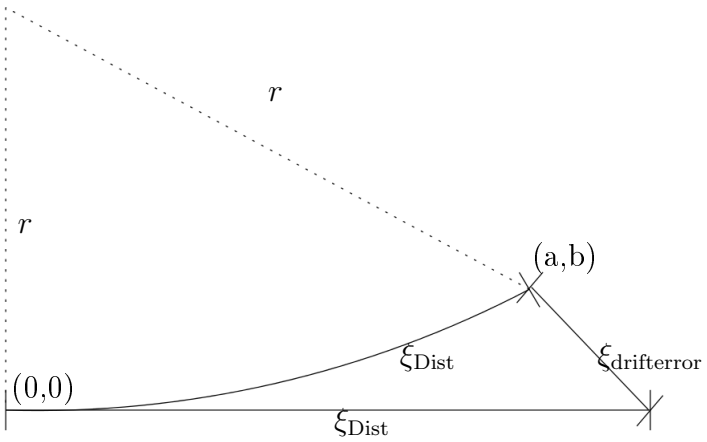


Figure 3.5: Circle radius and ξ relation.

The constraints we have are that the true path and the arc length should be the same, as both travel the same distance. The distance ϵ in Figure is equal to $\xi_{\text{drifterror}}$. Then the size of the circle is proportional to the angular drift, where 1 rotation through the circle would represent a, accumulated drift of 2π rad after travelling the distance $2\pi r$ then the drift per meter is $\frac{2\pi}{2\pi r} = \frac{1}{r}$. The relation between radius and arclength with the two points is given by

$$\xi_{\text{Dist}} = r \cdot \theta, \tag{3.1}$$

$$r \cdot \sin(\theta/2) = \sqrt{(a^2 + b^2)} \Rightarrow, \tag{3.2}$$

$$\sin\left(\frac{\xi_{\text{Dist}}}{2r}\right) = \frac{\sqrt{a^2 + b^2}}{2r}, \tag{3.3}$$

then we can solve for radius r by solving the system of equation

$$\frac{\sqrt{a^2 + b^2}}{2r} = \sin(\frac{\xi_{\text{Dist}}}{2r},) \tag{3.4}$$

$$a^2 + (b - r)^2 = r^2, \tag{3.5}$$

$$(\xi_{\text{drifterror}})^2 = (c - a)^2 + b^2, \tag{3.6}$$

which is solved with Matlab implmentation of Trust-Region Dogleg method[30].

RMSE error

Two different RMSE metric are used for estimating the RMSE. One RMSE metric is comparing the generated 3D map by the ICP-SLAM algorithm with a ground truth if it is available. Second RMSE metric is comparing the estimated travelled path with a reference path. Both the estimated travel path and the reference path are converted to a point cloud where ICP algorithm is applied to determine the RMSE value.

3.2.1 Test data 1: Forest

A Riegl scanner was used to determining a ground truth for the forest data set. The 3D map generated by Riegl and the one by ICP-SLAM method were downsampled using the

multi station adjustment (MSA) algorithm [1] which among other things removes noisy points such as small branches and even out the point density.

Since no detailed map of the area is available, the positional and angular drift error were not estimated. To compare the travel path, the position from the ICP-SLAM algorithm were compared to a visual based SLAM system [18] (VIS-SLAM). The result from this comparison were mainly used to check if the resulting path is reasonable. Loop closure error is still estimated and an example can be seen in Figure 3.6.

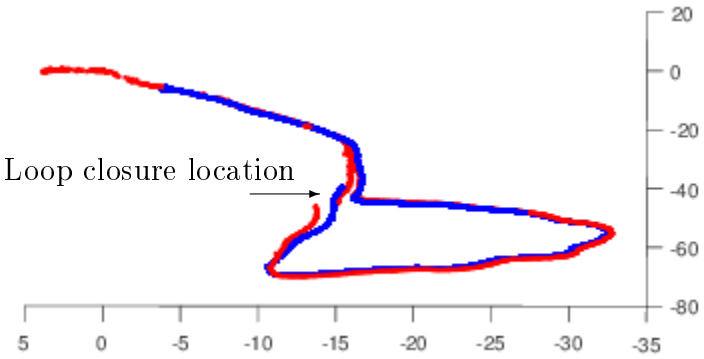


Figure 3.6: Example of loop closure error. Blue dots are generated from the visual slam system, and red from a sample run.

3.2.2 Test data 2,3: Urban

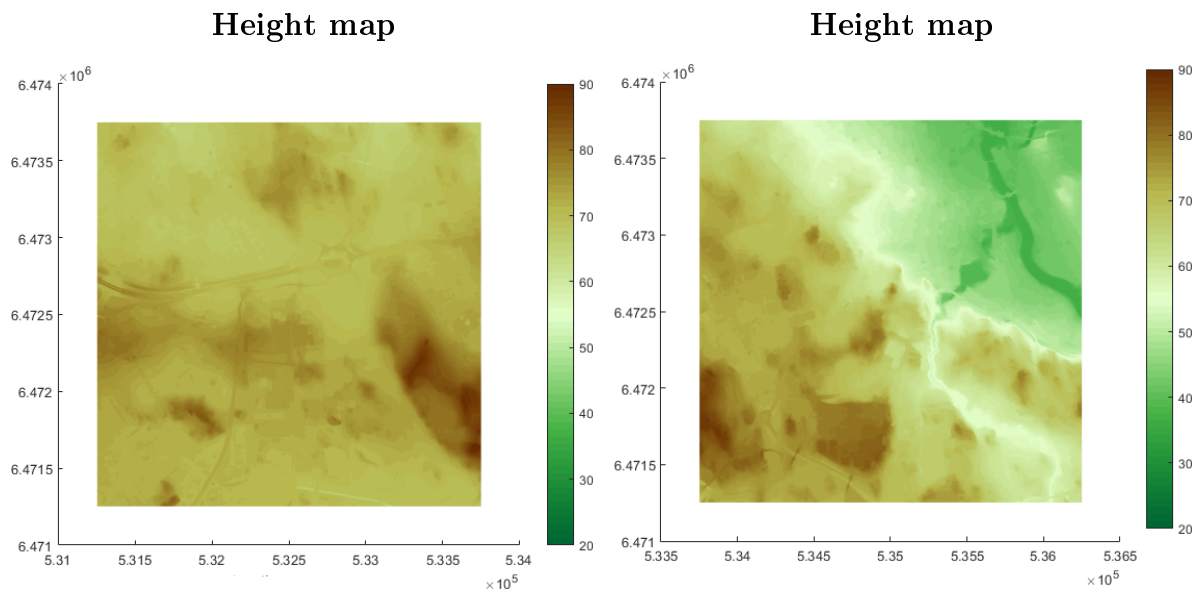
Two data sets were generated from the car mounted test, first one (test data 2) is refereed as the small urban data set, and the second (test data 3) is refereed as the large urban data set.

No ground truth is available for the measured urban environment, thus we could not compared to a reference model. But the pose location error can be estimated (see above) described in Section 3.2. By using data map from 'Lantmäteriet' at [19] in combination for a height map named 'GSD-Höjddata, grid2+' [20][2] reference points in 3D can be estimated which approximately represents the true travel path.

The points used to determine the expected travel length are marked with blue color, in Figure 3.8 for the short loop and Figure 3.10 for the larger loop. Sampled points for pose position are marked with blue circle and the travel path in a light blue line, which can be seen in Figure 3.9 for the short loop, and for the large loop in Figure 3.11. The height map can be seen in Figure 3.7

[1] Using Riegl scanner provided software.

[2] which is not publicly available, public version but with much lower resolution (30×30 meter squares) is available from geonames.



A. grid id: 64725 5325 25. *B. grid id: 64725 5350 25.*

Figure 3.7: Height map of grid2+ (Precision is limited to one height value per square of size 2×2 meter).

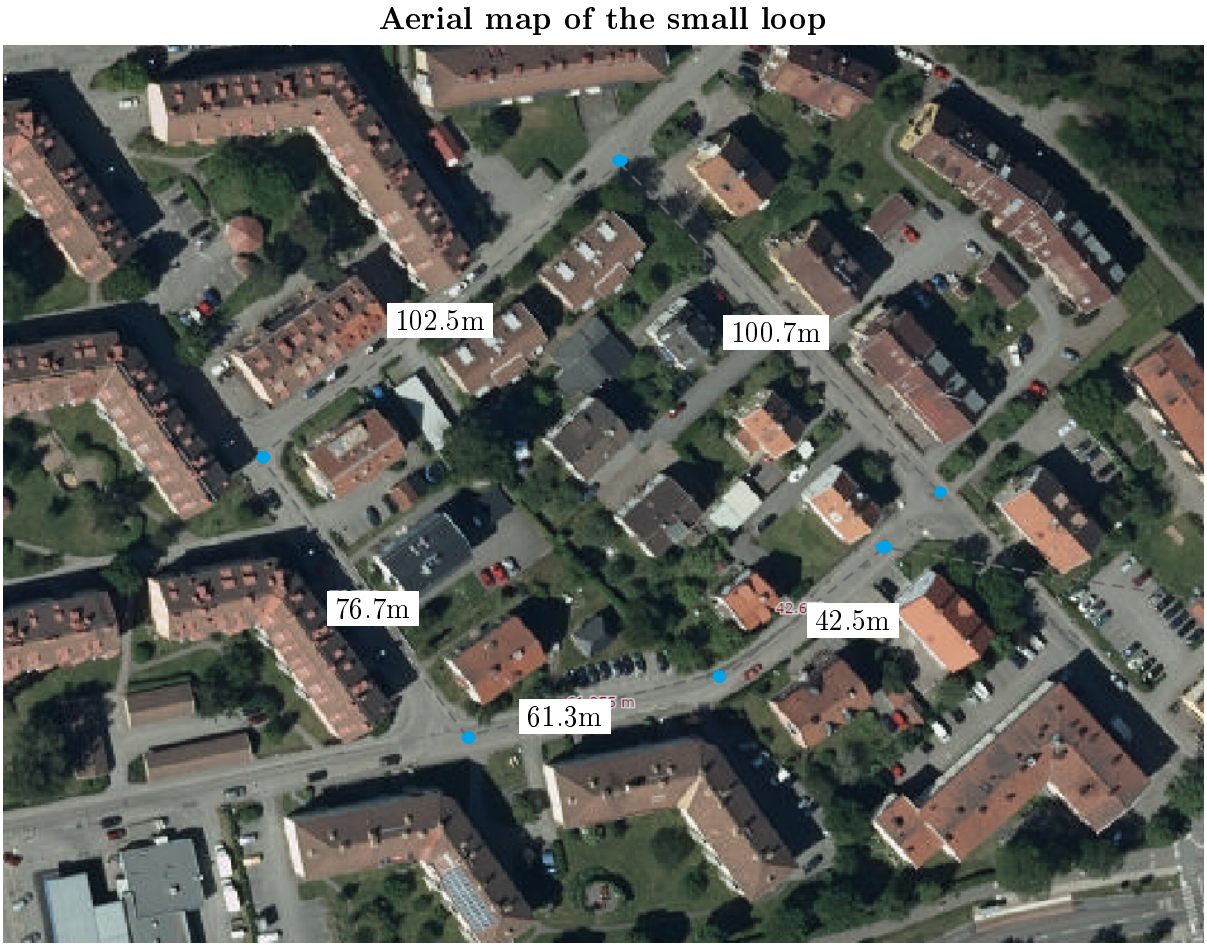


Figure 3.8: Blue points are key points used to determine the expected travel distance.

Topographic map of the small loop travel path

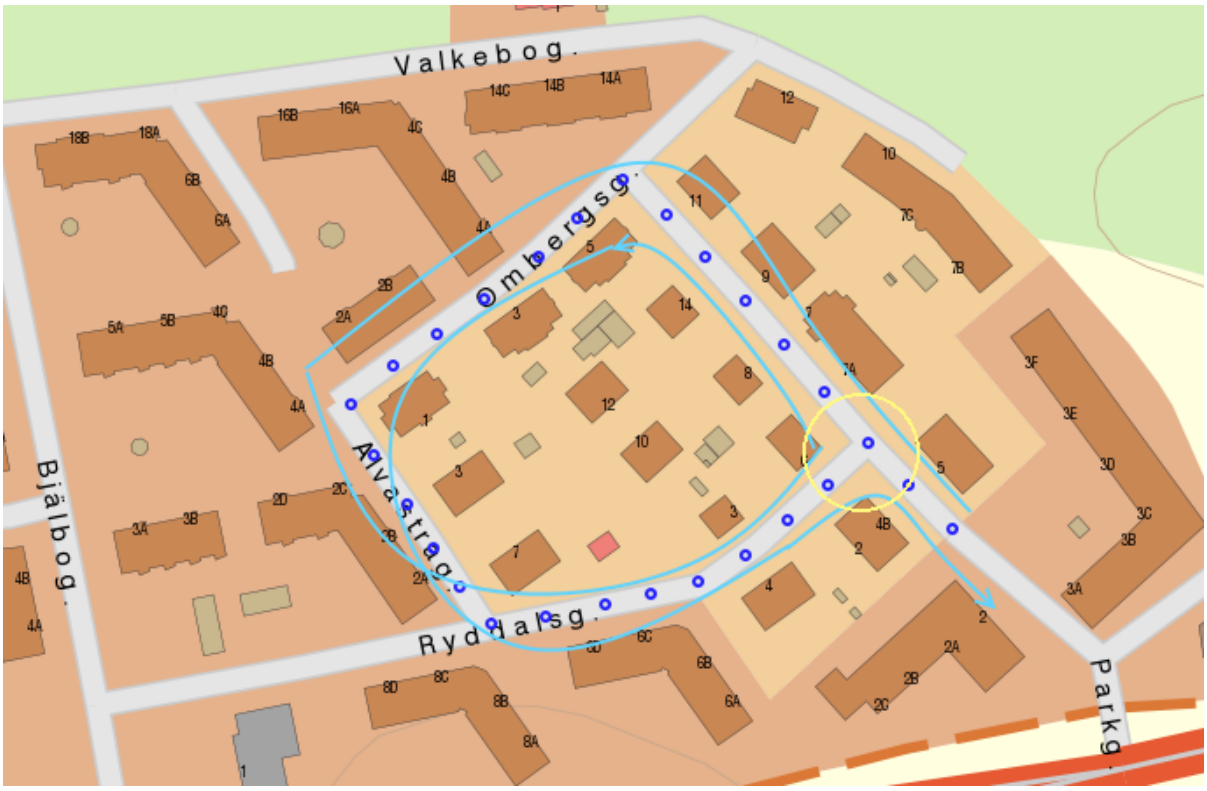


Figure 3.9: The blue circle are points which are used as a reference true positioning which is used to align against the SLAM-GICP position poses. Light blue color represent the path driven during the mapping. Yellow encircled area is used to evaluate the drift error as it corresponds to loop closure.

Aerial view of large loop area

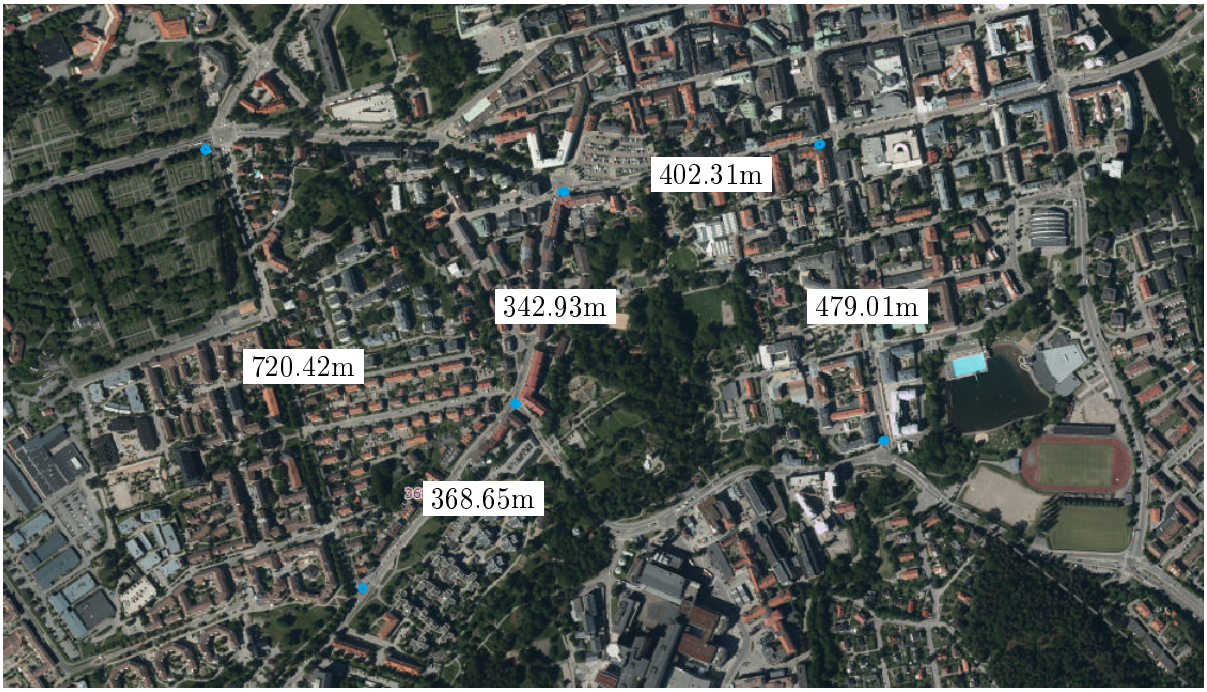


Figure 3.10: Blue points are key points used to determine the expected travel distance.

Topographic map of the large loop travel path

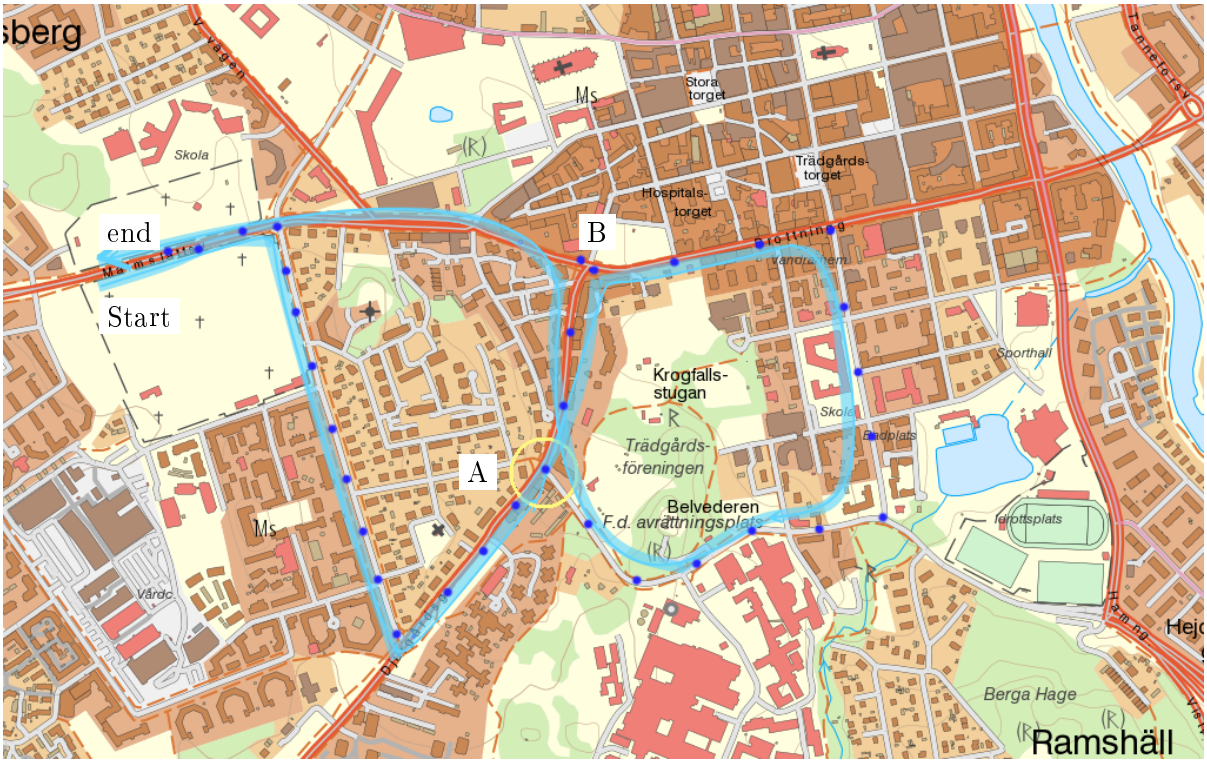


Figure 3.11: The blue circle are points which are used as a reference true positioning which is used to align against the SLAM-GICP position poses. Light blue color represent the path driven during the mapping. Yellow encircled area is used to evaluate the drift error as it corresponds to loop closure.

3.2.3 Test data 4: Countryside

For the countryside measurement, an evaluation if EKF can help is done, and if the 3 Hz IMU data can be utilized. The reference flight path was determined in the Mission Planner software which uses EKF with data from barometer, GPS, magnetometer, two sets of gyroscopes, accelerometer, and user input, which generates the reference path is shown in Figure 3.12.

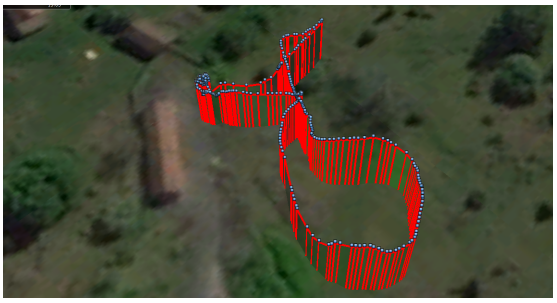


Figure 3.12: The estimated flight path from mission planner mapped onto google earth.

3.3 GICP-SLAM

For this section, the implementation of the SLAM method is described, where we begin with motivation of the approach and then how the motion model and how it is updated with a pose graph to store the poses. Details of cloud registration follows and this section concludes with the proposed sequence optimization.

3.3.1 Motivation

Based on the literature studies most of the methods involve storing observed features in an EKF and applying cloud features to cloud features matching based on that. This has a problem of having a complexity with grows quadratic with number of observed features. To solve this an ICP based method is used to solve point cloud to point cloud comparison . By preliminary investigations in article [32], the generalized-ICP algorithm is shown to have superior performance when working with sparse data and as such is chosen as our registration method (section 2.1.2). The basis for our state estimation will still be an EKF as it is flexible such that we can update the state from multiple types of input. EKF does also estimate the state uncertainty which will be useful for altering the point cloud registration.

In addition to EKF as state estimation a graph is created where each vertex is a pose (position and angle) of a LiDAR measurement and an edge is a relative position between two poses. This is done to investigate if it is possible to optimize the pose graph based on IMU data. As the thesis is more focused on how IMU can assist the SLAM process, less details goes into the graph optimization unless IMU is used to help.

A summary of the presented algorithm can be seen in Algorithm 1.

Algorithm 1 Overview of GICP-SLAM algorithm.

```
procedure GICP-SLAM(3D measurement source, IMU source)                    ▷
    Instantiate initial state and set unknown state to zero.
    'currentTime' = 0
    while Measuring do
        if Use a model then
            while 'currentTime' < 'nextLidarMeasurementTime' do
                Propagate model forward until next measurement and update current time.
                if Reaches IMU measurement before LiDAR measurement then
                    Update EKF with IMU observation
                    Read next IMU measurement and its timestamp
                end if
            end while
        end if                    ▷ Now 'currentTime is equal 'nextLidarMeasurementTime'
        Store latest Lidar measurement into 'sourceCloud'
        Preprocess 'sourceCloud' as in Section 3.3.3
        Get a 'setOfTargetCloud' as in Section 3.3.3
        parfor targetCloud in 'setOfTargetCloud' do
            Compare 'SourceCloud' with targetCloud                  ▷ Algorithn 2
        end parfor
        Store the result in a graph as in Figure 2.6
        Correct the EKF model with observation from GICP          ▷ Section 2.2.2, 2.2.2
        if Using LSO and criteria is fulfilled then
            Perform Local Sequence Optimization                     ▷ Section 2.4
        end if
        if Using graph optimization and criteria is fulfilled then
            perform graph optimization                              ▷ Section 3.3.4
        end if
    end while
end procedure
```

3.3.2 State estimation and update

The state estimation is done by setting up a motion model as described in theory section 2.2 where we can make a prediction of our pose in 6DoF while also being able to utilize IMU measurements. The initial state is set as zero but for the roll and pitch angle are estimated from the equations

$$\mathrm{q_{pitch}} = q_x = \arctan(\frac{a_y}{a_z}), \tag{3.7}$$

$$\mathrm{q_{roll}} = q_y = \arctan(\frac{-a_x}{\sqrt{a_x^2 + a_y^2}}), \tag{3.8}$$

with accelerometer data sampled close to the first available LiDAR measurement. The local coordinate system is set to $e_{i\ local} = R_z R_y R_x \cdot e_i$.

If IMU data is available, the first possible improvement to the SLAM process is to update the EKF with IMU observations as described in Section 2.2.1. This is done as one purpose was to investigate whether or not IMU data can improve the SLAM process by contributing to a better state estimation. As point cloud registration often has multiple local minima, an a better prediction should yield a result ending up in a better local minima. Updating the EKF based on registration results are done according to the theory in Section 2.2.2.

To be able to update the state via EKF some noise values have to be approximated. The model noise values used in Section 2.2 are set as $\epsilon_{acc} = 0.5^2$ and $\epsilon_{angvel} = 1.0^2$. The noise of IMU measurement where set as $\sigma_{acc} = \sigma_{angvel} = 0.08^2$ and were estimated by noting the fluctuation when laying still. Noise value used for the point cloud registration were set to $\tau_{pos} = \tau_{ang} = 0.001^2$ and were chosen by varying the process and observation noise ratio which yielded reasonable results (details in result Section 4.1.2).

3.3.3 Point cloud registration

Preprocessing measurement

Due to the method Velodyne scans the surrounding, we will have to compensate for the motion blur effect which occurs. We will also have to remove static points which can potentially affect our point cloud registration.

Motion blur is dealt with by assuming a constant motion during a revolution, and each lasers records at the same time for every azimuth angle. Then there is a transformation matrix which takes the pose at the end of scan to the pose at start of scan. Each azimuth angled scan is then transformed as a ratio of the transformation matrix to reduce the influence by motion blur.

The static measurement is filtered out by assuming that if a certain azimuth angle and elevation angle in recent scans yields the same time of flight, then it is assume that the point is a static one and those distances is then marked as 'NaN' (not a number). A matrix is used to keep score of each azimuth and elevation angle combination, which holds values between 0 and 25, if a score is higher than 20 it is assumed to be a static point. The score is increased when similar time of flight is detected, otherwise it is decreased.

Selection of target set

The latest measurement is compared to a target set which consists of a number of recent frames and key frames. Key frames are frames which have been compared to a larger set of frames. A frame is marked as key frame if the accumulated travelled distance since last key frame is larger than 1 meter, or the accumulated angular movement is more than 0.1 radian.

The the latest measurement is marked as key frame, the target set consists of 6 most recent frames and 6 key frames. Else it consists of 4 most recent frames and 2 key frames.

GICP preprocessing

The initial guess for the pose is taken from the motion model and then both target and source cloud are translated such that the source cloud camera is located at origin. This is done such that any rotation applied is centred around pose center. This is important as the minimal angular rotation step is fixed and if the cloud is far from origin then a small angular rotation corresponds to a locally combination of a large translation and small rotation.

Quad mesh are then generated for both clouds, which is based on the 2D-manifold described in Section 2.1.2. For the VLP 16 based data a quad mesh is generated by 3 points in horizontal direction and 2 points in vertical direction. HDL 32 based data uses 5 points in horizontal direction and 3 points in vertical direction. Then a normal can be approximated flipped against the viewpoint direction.[3]

The covariance of each point is then determined by the method described in Section 2.3 with the uncertainty of the normal direction set as $1e^{-4}$. If the alternative surface is to be used, then it is determined as described in Section 2.1.2.

GICP algorithm

The parameter used for the GICP algorithm was, 1.5 meter as the threshold of finding nearest neighbour, a solution is found if the distance between two iteration were less than 0.005 meter and angular rotation of less than 0.001 radian. The GICP algorithm were allowed to run for 20 iteration while the inner BFGS algorithm which solves the optimization function were allowed to run up to 10 iterations. Rest of the parameter not mentioned here we used as the default values[4].

As we have access to more information than just two general clouds, such as a state prediction uncertainty from our EKF filter, an alternative cost function is proposed in Chapter 2.3 Equation (2.65) which is referred as the modified GICP (mGICP). This state prediction also includes a state estimation error in the form of covariance P (Section 2.2) which we can use. By calculating the invers of P and setting $M = P^{-1}$ where M is the M matrix in Equation (2.65) we can penalize solution which is far from our estimated state, the more confidence we are in our solution, the more penalize those result.

Using IMU for the EKF could help in the mGICP cost function as it would give us a better estimate on the certainty of our predicted state and mGICP is one of the area investigated to adjust the cost function. An additional limit it set such that confidence (standard deviation) on position is never lower than 0.05 m and the confidence in angles

[3]Other settings used are the default settings in the implementation in PCL 1.8.0 library.

[4]Implementation by the default EIGEN library BFGS solver in C++, and GICP in PCL 1.8.0 library.

Algorithm 2 GICP point cloud registration.

```
procedure CLOUD TO CLOUD MATCH(CloudPairsToCompare,Covariances)
    parfor Every pair of comparison do
        while Iteration < maxOuterIteration and not converged do
            for Each point in SourceCloud do
                Update KD-tree, insert the target points
                for each point in SourceCloud do
                    Find nearest point in the KD-tree within threshold
                end for
                Determine the cost function and derivative according to section 2.1.2
                while Iteration < maxInnerIteration and not converted do
                    Apply one step BFGS alghorithm
                    Check for convergence
                end while
                Check for convergence
            end for
        end while
    end parfor
return All possible transform, number of pairs, cost
end procedure
```

were never 2.5 degrees, this is done so that GICP is never forced to use a prediction which could lead to a positive feedback loop where cloud registration does not provide any new information.

Post processing of point cloud registration

As multiple comparison are done and there is a potential for bad matches, and as such the cloud registration result is filtered based on resulting cost function and number of matched pairs. A filter is used to remove bad matches according to the criteria

$$n_{pair} > (0.4 + (s_d \cdot t_d + 1) * 0.22) \cdot m_{pair} \tag{3.9}$$

$$n_{value} < 2 * m_{value} \tag{3.10}$$

where n_{pair} is the number of matched pairs and n_{value} is the resulting cost value from GICP. m_{pair} and m_{value} is the median numbers of pair and median function value of the 10 most recent chosen comparison. s_d is the direction of the source pose and t_d the direction of target pose. The dot product is to compensate for uneven point density due to the camera being tilted or if section of the LiDAR measurement is cut off.

Those clouds which pass the filter have their transformation added as an edge in the pose graph. The one with the lowest value function is then used to update the EKF. If no matches passes the filter, no update is given to the EKF.

3.3.4 Sequence optimization

The sequence optimization algorithm is called if it has been more than η_{freq} since last time the heuristic were used or if a loop closure is detected and graph optimization is

enabled. The basis of using LSO is that it need information for some state which have an error independent of the time of measurement. For loop closure, the pose can be acquired, but from internal sensors only the roll and pitch can be estimated. [5]

Local Sequence Optimization with IMU data

From the IMU we can estimate the roll and pitch angle and it is sampled by the usage of the complementary filter described in Appendix C. The resulting pitch and roll is given the variable name π_{pitch}, π_{roll}. It relies on yaw estimate from the EKF which has a bias from the GICP measurement.

As it only estimates two (roll and pitch) out of the six states in a pose we have to estimate the remaining four. Those can either be taken from the last pose or the states in the EKF filter [6]. The values which are used to fill in the missing states are given by $\kappa_x \kappa_y \kappa_z \kappa_{yaw}$. They are then given a lower weight η_{weight} as they are less desired to optimize after, but preferably non zero because those values is close to the correct one.

Additional criteria for using LSO with IMU data were added, the accumulated angular movement had to be less than $\eta_{angstab}$ since last LSO so that it was not applied during a very noisy movement. If all criteria is fulfilled then the LSO algorithm in section 2.4 is applied with the target state as

$$E_d = [\kappa_x \ \kappa_y \ \kappa_z \ \pi_{pitch} \ \pi_{roll} \ \kappa_{yaw}]^T \tag{3.11}$$

and the cost matrix T as

$$T = \text{Diag}\{\eta_{weight}, \eta_{weight}, \eta_{weight}, 1, 1, \eta_{weight}\} \tag{3.12}$$

Values which seems to work well is $\eta_{freq} = 30$, $\eta_{dist} = 60$, $\eta_{weight} = 0.03$, $\eta_{angstab} = 0.2$ and to fill in the missing states from last pose.

[5] Even though they have a slight bias from yaw estimates from GICP, but it still works based on the results.

[6] The last pose is the value that were feedbacked into EKF, while the states of EKF is the state after the feedback.

Algorithm 3 Local sequence optimization algorithm.

```
1: procedure SequenceOptimization
2:     Determine the set of poses G in the pose graph whose edges can affect outcome.
3:     p = The pose to be optimized (most recent one)
4:     while Not converged and e_max is limited do    ▷ e_max is the allowed cost for swapping edges
5:         for each pose t in G do
6:             if t can affect the pose of p then
7:                 sort the edges of t after the GICP cost function value
8:                 for each edge between t and previous pose and edge < e_max do
9:                     if Check if new edge leads to a more desirable state for p then
10:                        Set optimization as not converged
11:                    end if
12:                end for
13:            end if
14:        end for
15:        Increase e_max or set it as unlimited
16:    end while
17: end procedure
```

Graph optimization

If graph optimization is enabled, it will solve the optimization function as in section 2.4. If frame i loop closes with frame j and triggers graph optimization, then in 3) line 2 will add poses into the set G which belongs to the shortest path between i and j. Those poses are within the blue shaded area in figure 3.13. Same algorithm but in line 9 the value function is adjusted based on section 2.4 and the poses it optimize after are the red and green edges in figure 3.13.

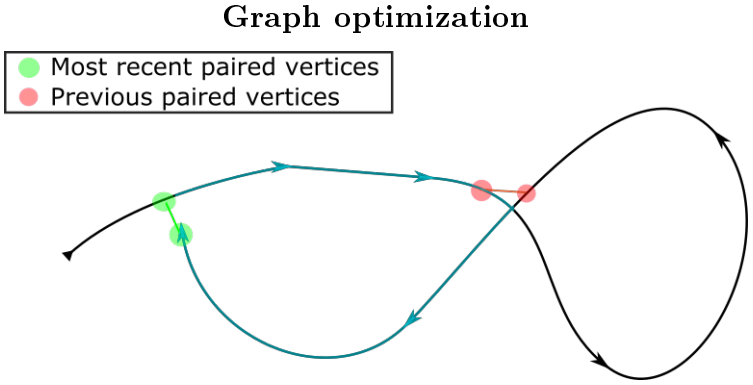


Figure 3.13: Black lines are the current pose path, the green shaded circle is the most recent detected loop closure with a green bar representing their relative position. Red shaded circle is previous detected loop closure location with the pose relative position. Vertices belonging to poses within the blue shaded line are those vertices where we can potentially change edges in our pose graph.

Chapter 4

Results

The results of the GICP-SLAM method are presented here. First section presents the results of varying parameter primarily onto the *test data 1 and 2*, with additional result available in Appendix D. The following section includes summary of result for each data set with distinct settings. In Appendix A the parameter values and settings used in the test (unless otherwise specified) are given.

4.1 Parameter variation

4.1.1 Surface approximation

In Figure 4.2 we can see the loop closure error for changing the surface approximation type which shows it having low impact on closure error. The effect on the pose path can be seen in Figure 4.1 where the blue line is typical result when using the 1-1 surface approximation which frequently mismatches and for the proposed proportional surface approximation the pose path remain reasonable throughout the run.

4.1.2 Process and observation noise

For the result of varying the process and observation noise can be seen in Table 4.1 where the model noise of EKF were fixed to $\epsilon_{acc} = 0.5^2, \epsilon_{angvel} = 1^2$ and the observation noise of IMU to $\sigma_{acc} = \sigma_{angvel} = 0.08^2$. The process noise of the GICP algorithm $(\tau_{pos}, \tau_{angle})$ were increased from 0.0 to 0.243^2 and for the small urban loop some extra runs were computed. Other settings were default once expect for the limit of optimization process were set to 40 outer iteration and 20 inner iteration. Observation noise level 0.001^2 and 0.027^2 results in stable solution.

4.1.3 Modified GICP

Alpha values which were investigated were $\alpha_{gicp} = 0.05$ to $\alpha_{gicp} = 0.8$ with a 0.05 intervals. The result of the variation can be found in Figure 4.3 and Appendix D.1. Only notable is for most α_{gicp} values the forest set improves while the urban set have limited effect.

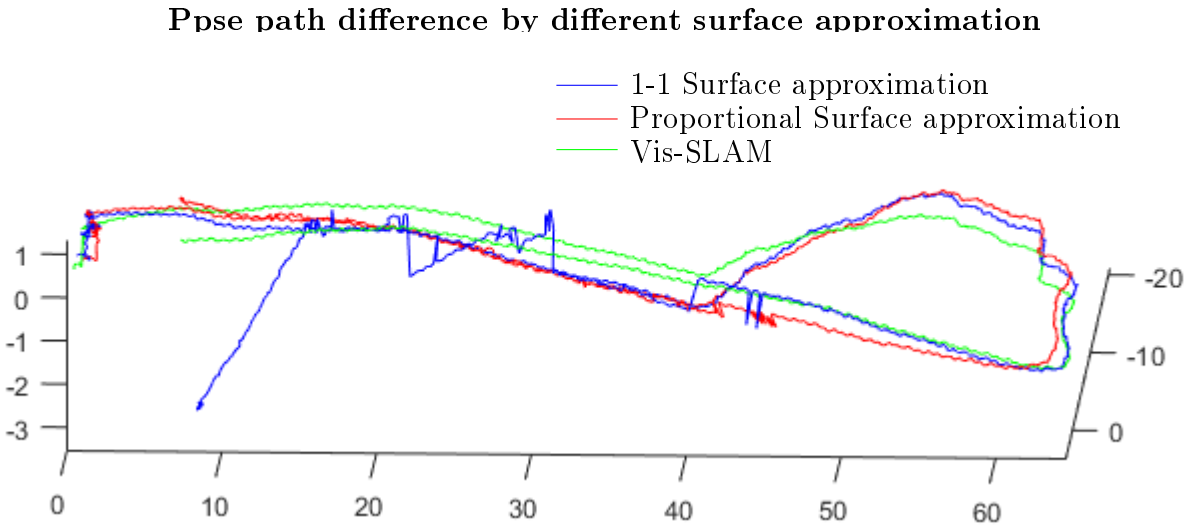


Figure 4.1: Blue line is a representative result of the pose path when using 1-1 Surface approximation with various settings. The red line pose trajectory result from using purposed proportional surface approximation. Green line is pose trajectory from the visual SLAM system.

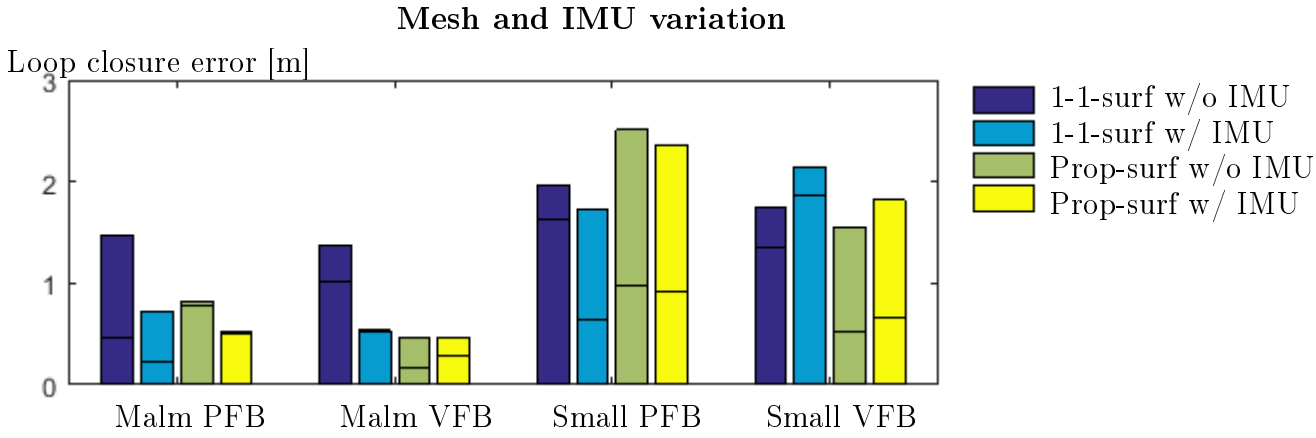
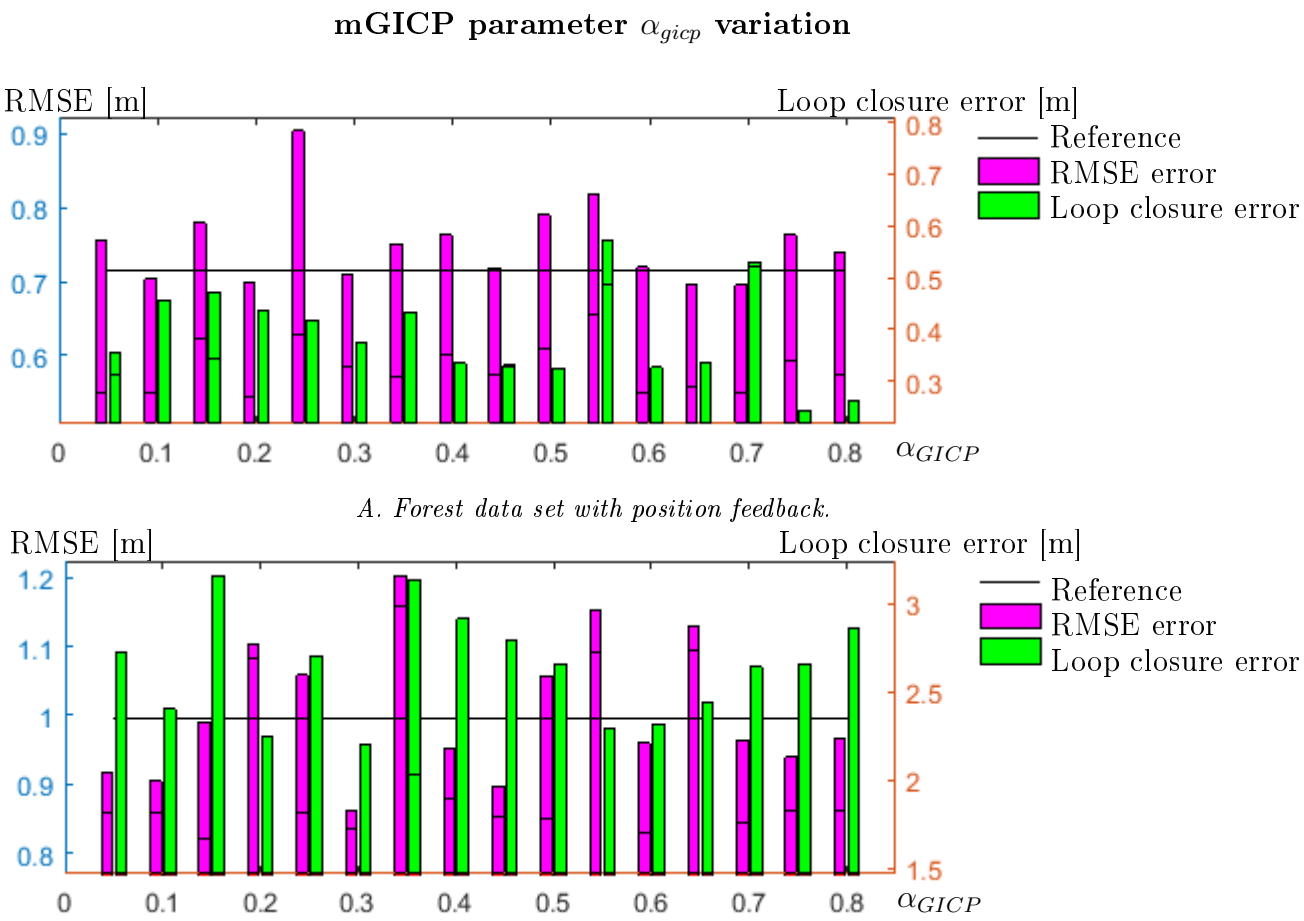


Figure 4.2: Surface comparison between 1-1 and proportional surface approximation. Area under the line is the loop closure error on Z-axis.

*Table 4.1: Process and observation noise ratio varied. Test were done on the small urban data set and the forest data. The evaluation metric were the size of loop closure error when returning to a previous position in meter. 'inc' for inconsistent result. * Inconsistent jump at start but the SLAM recovers.*

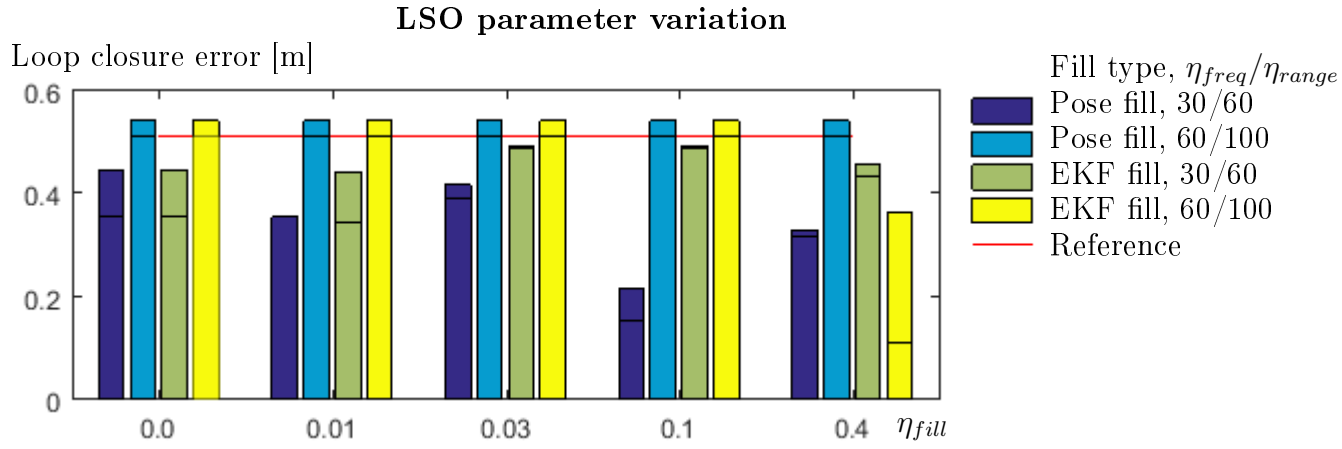
$\tau_{pos}, \tau_{angle} =$	0	10^{-6}	$9 \cdot 10^{-6}$	0.009^2	0.027^2	0.081^2	0.243^2	0.55^2	0.7^2	0.85^2
Urban, Pos Fb	2.17	3.07	2.31	2.22	2.70	2.58	0.54	1.01	1.09	inc
Urban, Vel Fb	2.41	2.30	3.29	2.24	3.33	2.06	2.96	n/a	n/a	n/a
Forest pos	0.51*	0.11	0.39	0.73	0.53	inc	inc	n/a	n/a	n/a
Forest Vel	0.94*	0.4	0.474	0.48	0.68	inc	inc	n/a	n/a	n/a



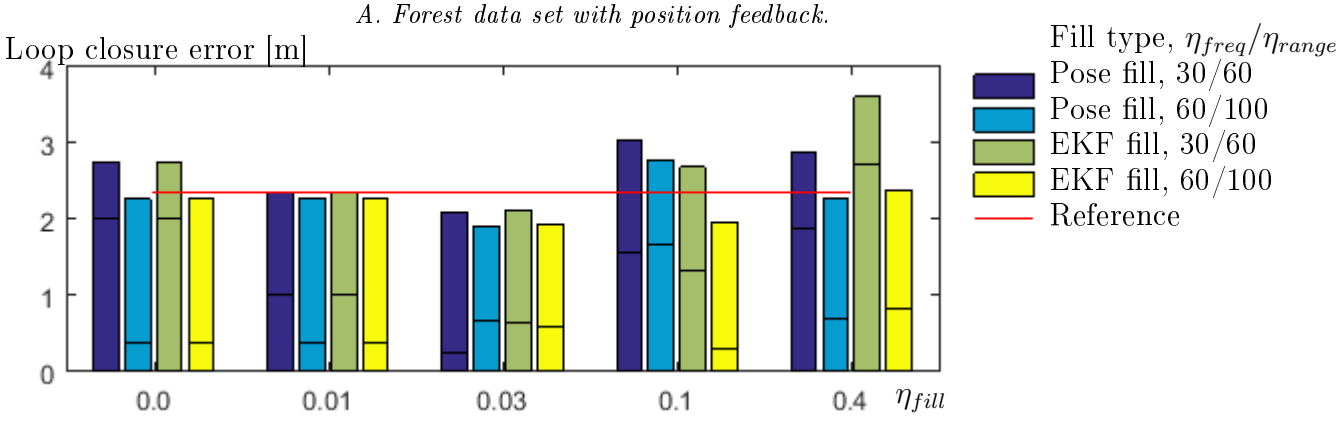
A. Forest data set with position feedback.

B. Small urban data set with position feedback.

Figure 4.3: The α_{gicp} parameter varied from 0.05 to 0.8 with 0.05 intervals applied on the forest and small urban data set with position feedback. The black line is the reference run with position feedback and IMU. Line over the magenta bar is the RMSE error when projecting the estimated path on XY plane. Line over yellow bar is the loop closure error located on z-axis.



A. Forest data set with position feedback.



B. Small urban data set with position feedback.

Figure 4.4: Parameter variation for the LSO applied on forest and small urban data with position feedback. Redline is the reference error for the case position feedback with IMU. The fill type as described in Section 3.3.4. Black bar is the loop closure error on Z-axis.

4.1.4 Local Sequence Optimization

For the LSO test, the parameter tested where $\eta_{freq} = 30, \eta_{dist} = 60$ and $\eta_{freq} = 100, \eta_{dist} = 100$, with the η_{weight} set as $0.0, 0.01, 0.03, 0.1$ and 0.4. In addition it was tested to use EKF or pose to fill in the unknown states. The results can be seen in Figure 4.4, but nothing significant except of $n_{weight} = 0.03$ can potentially improve the process.

4.1.5 Frame removal

Frame removal was applied on the small urban data set and the forest data set, where on the forest case the metric shows if the SLAM process finished or not (Table 4.2, and for the urban loop the metric were on which $\sim 90°$ turn the SLAM process failed to yield reasonable path (Figure 4.5).

Table 4.2: Only register every n:th point cloud on the forest data set, where pass is successfully generating reasonable 3D map.

Scenario \ $n =$	2	3	4
SLAM without IMU	Pass	Fail	Fail
w/ IMU	Pass	Pass	Fail
w/ LSO, mGICP	Pass	Pass	Fail

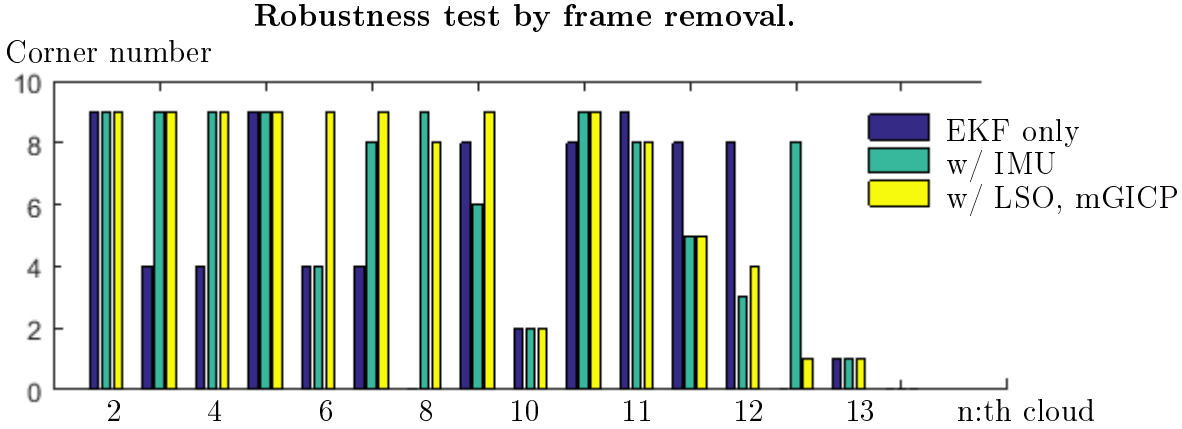


Figure 4.5: Only every n:th point cloud from the small urban data set were used for registration. The y axis is the corner number which the SLAM process failed, with number 9 as completed the whole data set. Blue bar is the result of only using EKF with position feedback with default parameter values (Table 4.4 run 2). Green bar is the result when also using IMU to improve the motion model (Table 4.4 run 3). Yellow is the result from using IMU with LSO and mGICP (Table 4.4 run 9).

4.2 Test data 1: Forest data set

Bar graph in Figure 4.6 displays the loop closure error and RMSE value when compared to the VIS-SLAM system. The settings for each run in Table 4.3. Overall slight improvement can be seen by utilizing IMU data but improvement from randomness cannot be ruled out.

Top view of pose path generated by run 2 and 11 with Vis-SLAM in Figure 4.7:A and zoomed in at the loop closed area in Figure 4.7:B. Overview of cloud generated by run 2,11 and Riegl in Figure 4.9, and side view in Figure 4.10, and a tree Figure cut in 4.8.

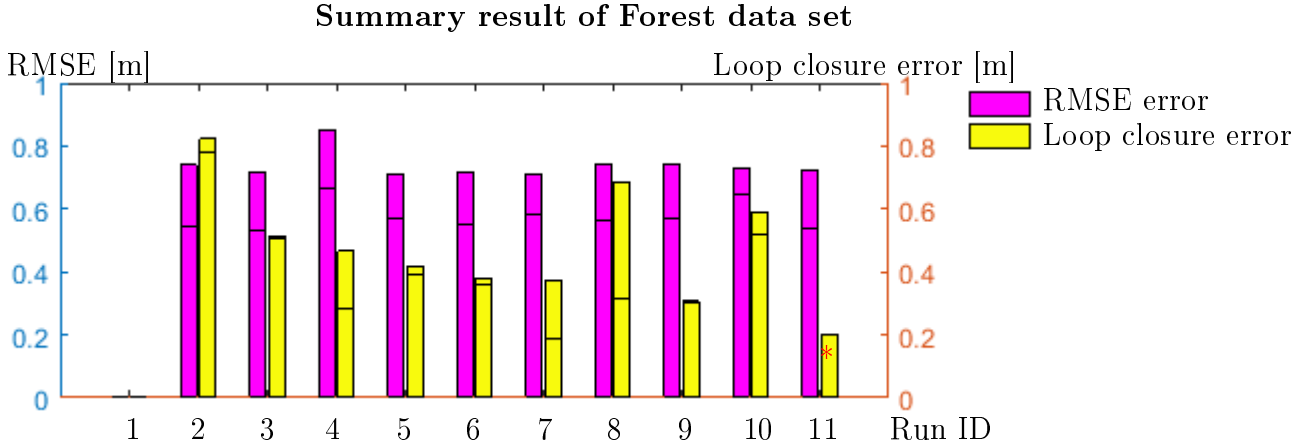
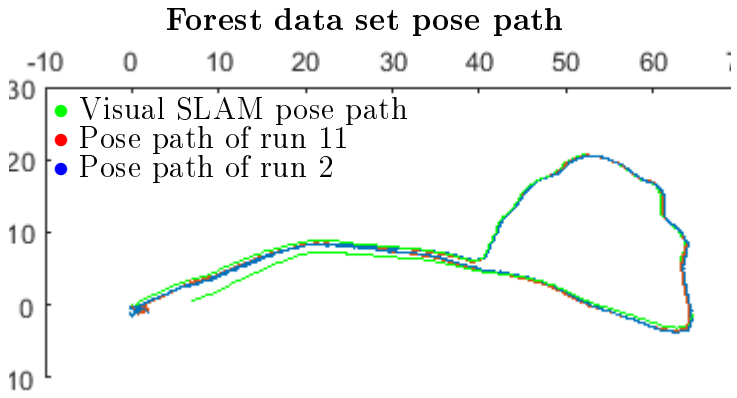


*Figure 4.6: Dash in magenta bar is the RMSE when projecting pose path onto the XY plane. Dash on yellow bar is the error in Z-axis. * loop closure error smaller than 0.2 . Additional run settings details in in Table 4.3.*

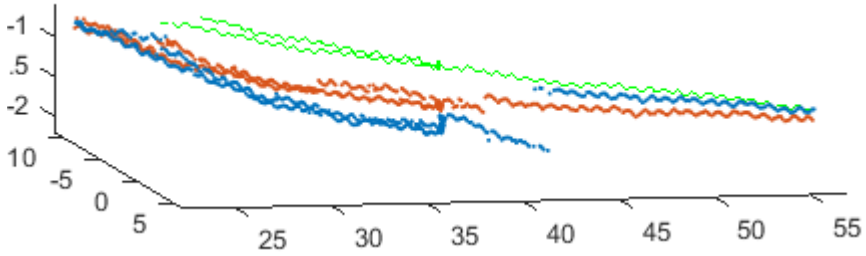
Summary result of Forest data set

Table 4.3: Summary of results for various settings with default parameter applied on the Forest data set. LSO parameter were set to $\eta_{freq} = 30, \eta_{range} = 60$. Corresponding bar graph in Figure 4.6. LCE is loop closure error.

Run	Settings		LSO	mGICP	LCE	LCE	RMSE	Notes
	IMU	FB	η_{weight}	α_{gicp}	$[m]$	$[rad]$	(Riegl)	
1	No	$P, \tau = 0$	-	-	n/a	n/a	n/a	Inconsistent
2	No	P	-	-	0.82	0.034	0.5513	
3	Yes	P	-	-	0.52	0.047	0.5579	
4	Yes	V	-	-	0.46	0.044	0.6804	
5	Yes	P	0.03	-	0.42	0.035	0.4825	
6	Yes	V	0.03	-	0.38	0.046	0.4912	
7	Yes	P	-	0.3	0.37	0.053	0.6050	
8	Yes	V	-	0.3	0.68	0.072	0.5621	
9	Yes	P	0.03	0.3	0.31	0.051	0.5554	
10	Yes	V	0.03	0.3	0.59	0.025	0.5948	
11	Yes	P			<0.2	<0.01	0.4574	Graph OPT



A. Pose path from a top view.



B. Side view of the pose path at loop closing area.

Figure 4.7: The pose path is plotted of run 2, 11 and the Vis-SLAM system, axis unit is given in meter.

A. Tree cut from Riegl scanner. *B. Tree cut from run 11.* *C. Tree cut from run 2.*

Figure 4.8: Comparison of a tree cut of the same tree from different runs.

Part of 3D map by forest measurement.

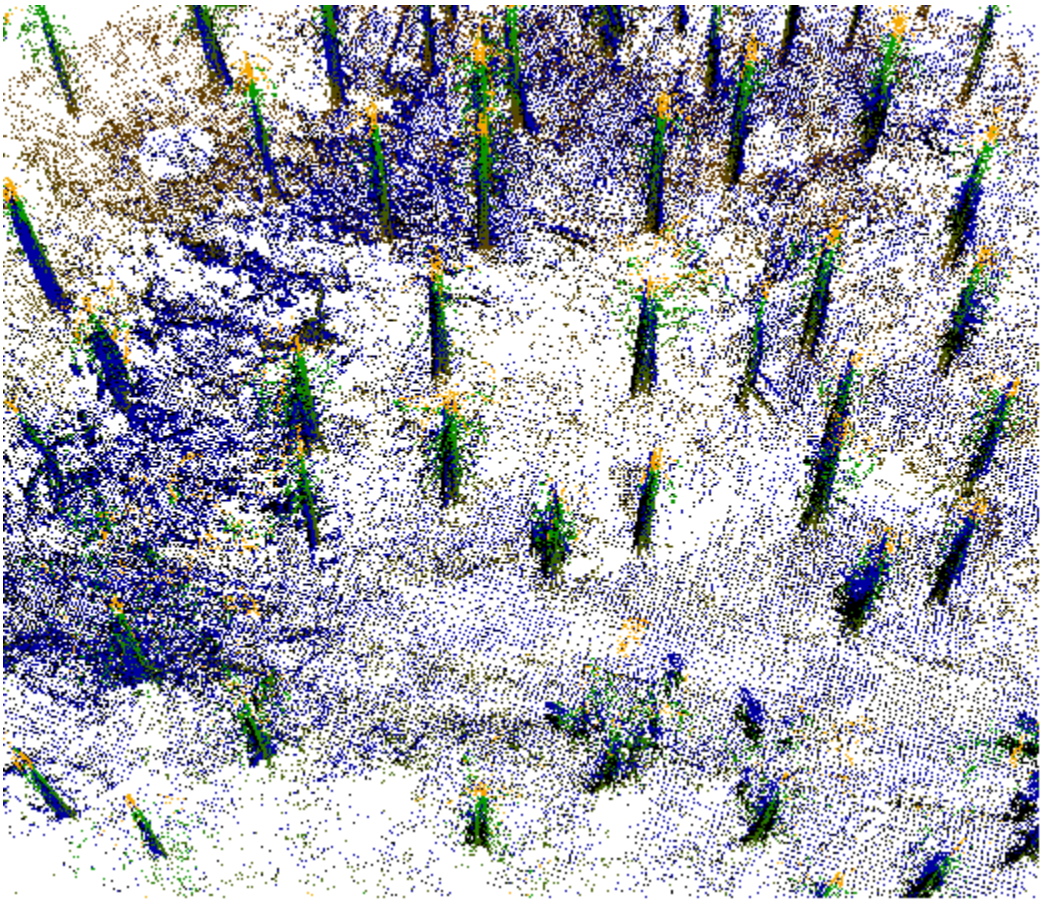
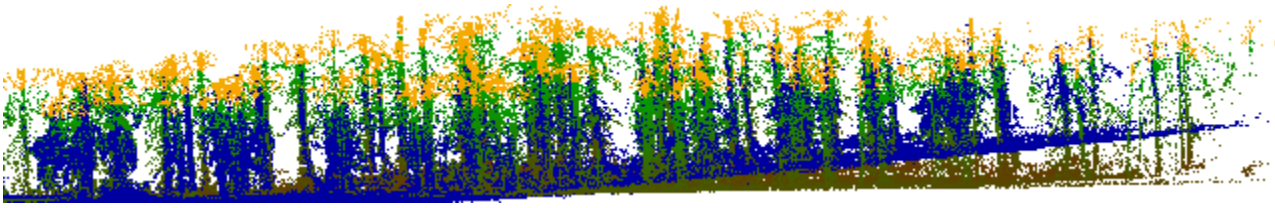
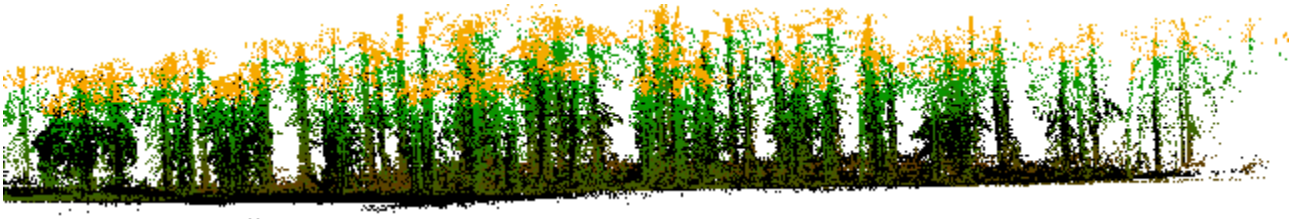


Figure 4.9: Overview of run 2, 11 and Riegl after MSA downsampling is applied. Green-Yellow scaled points are from Riegl scanner, Blue from run 2 and black from run 11. The pillars are trees.



A. Run 2 (blue) compared with Riegl (Green-yellow scaled).



B. Run 11 (black) compared with Riegl (Green-yellow scaled).

Figure 4.10: Side view comparison of forest data set.

4.3 Test data 2: Small urban data set

Bar graph in Figure 4.11 with the RMSE error compared with reference points and the loop closure error. Table with run settings and more detailed results in Table 4.4. Overall the results is neutral with no significant improvement by using IMU.

Pose graph of the run 1,4,11 with reference points compared in Figure 4.12 where a top view and a tilted view of the poses. Cloud generated by run 4 and 11 can be found in Figure 4.13, where overview of the result can be found and a more detailed view of the loop closure area.

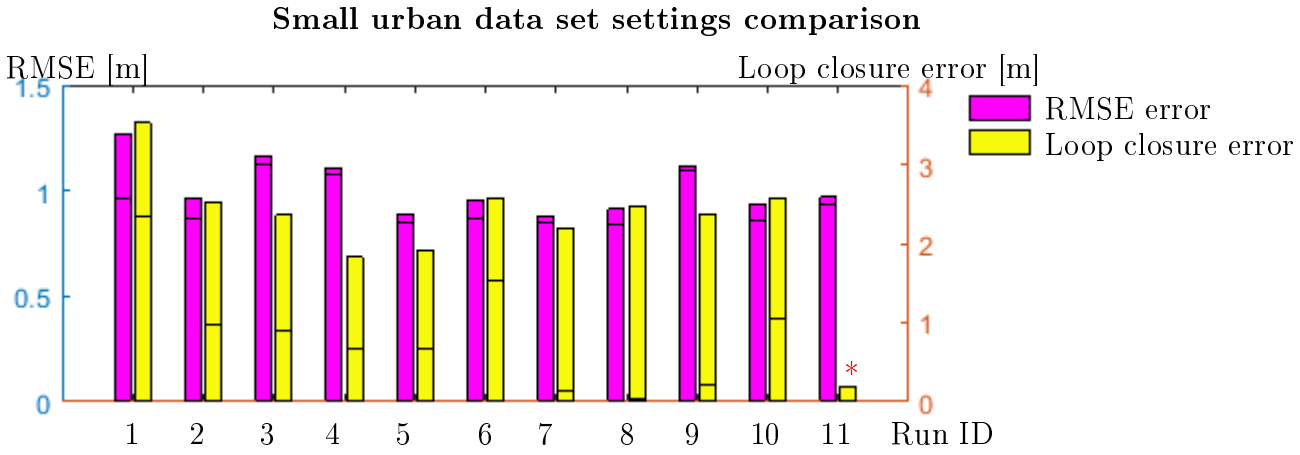


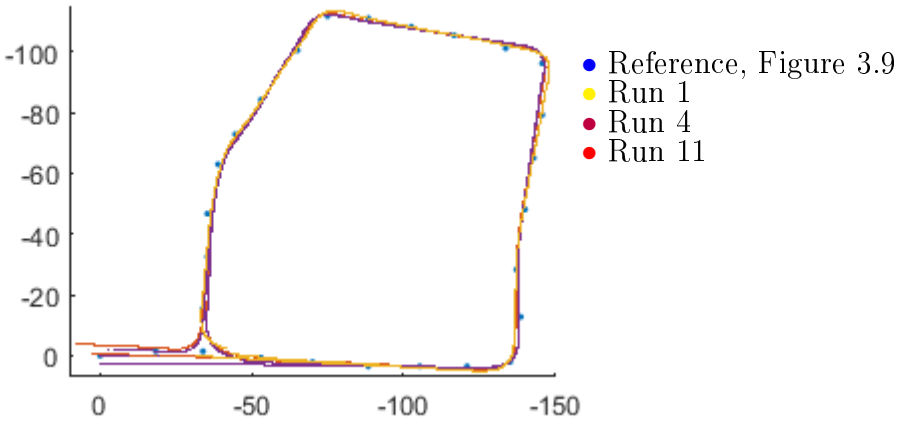
*Figure 4.11: Dash in magenta bar is the RMSE error when projecting pose path onto the XY plane. Dash on yellow bar is the error in Z-axis. Additional details in Table 4.4. * loop closure error less than 0.3 m.*

Comparison of varied setting on test data 2

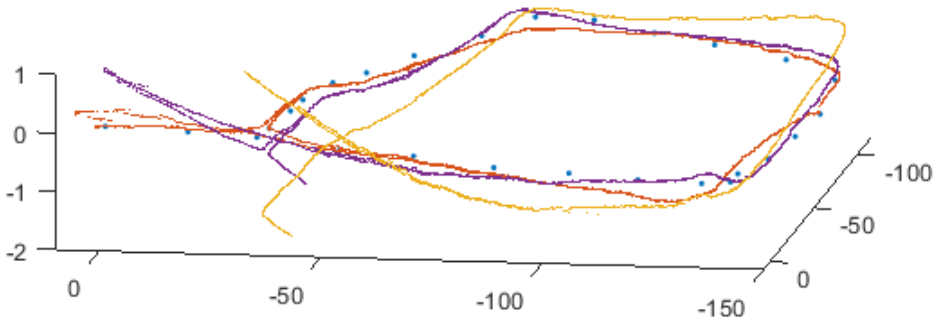
Table 4.4: Summary of results for various settings with default parameter applied on urban data set. LSO parameter were set to $\eta_{freq} = 60, \eta_{range} = 100$. Corresponding graph in Figure 4.11. LCE is loop closure error

Run	Settings		LSO	mGICP	LCE	LCE	Drift	Drift	Notes
	IMU	FB	η_{weight}	α_{gicp}	[m]	[rad]	$10^{-3}\frac{[m]}{m}$	$10^{-5}\frac{[rad]}{m}$	
1	No	P	-	-	3.52	0.047	2.1	3.7	$\tau = 0$
2	No	P	-	-	2.51	0.039	1.7	2.5	
3	Yes	P	-	-	2.36	0.043	1.7	2.3	
4	Yes	V	-	-	1.83	0.045	1.8	2.5	
5	Yes	P	0.03	-	1.92	0.053	1.2	1.9	
6	Yes	V	0.03	-	2.56	0.049	2.1	2.3	
7	Yes	P	-	0.3	2.20	0.047	2.2	2.3	
8	Yes	V	-	0.3	2.47	0.047	3.0	1.7	
9	Yes	P	0.03	0.3	2.37	0.048	1.1	2.6	
10	Yes	V	0.03	0.3	2.57	0.057	1.6	2.7	
11	Yes	P	-	-	<0.3	<0.01	-	-	Graph opt
-	No	V	-	-	1.54	-	-	-	

Pose path for small Urban set



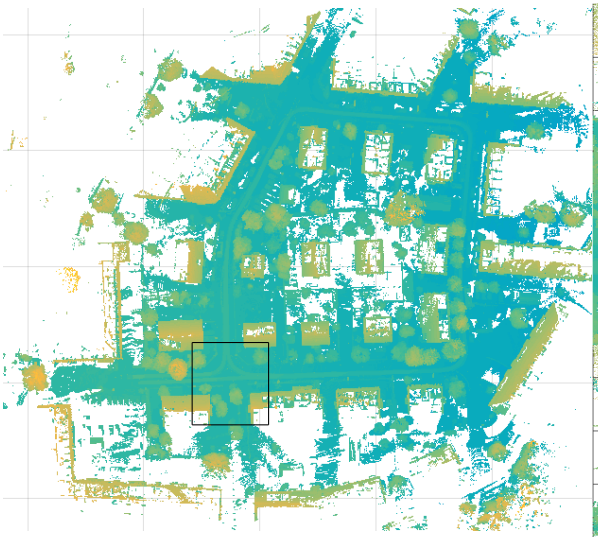
Top view of three pose paths.



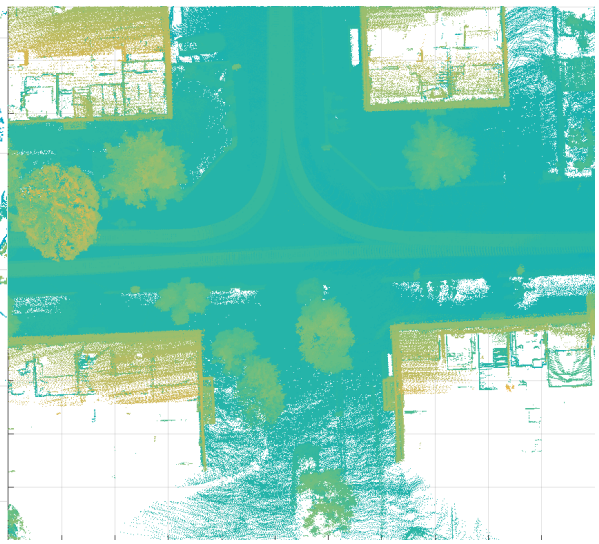
Side view of three pose paths.

Figure 4.12: Posepath comparison between run 1,4,11 with reference points in blue.

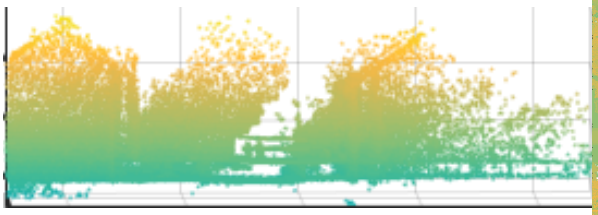
3D map results over small urban data set



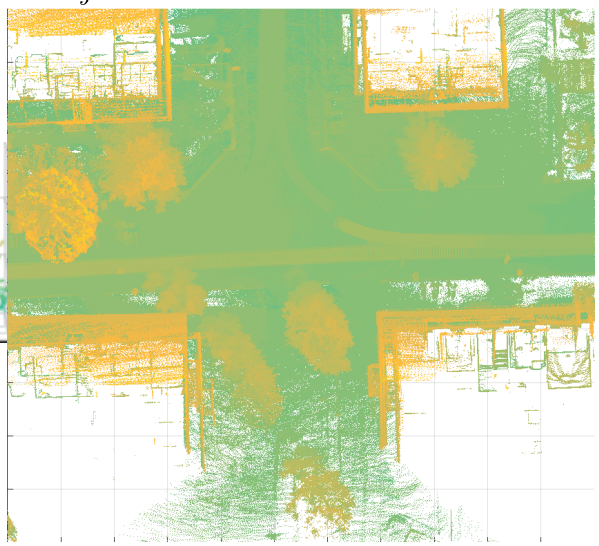
A. Overview of result from run 11 and the marked area is where the loop closes.



B. Enlarged area of the marked one in A, by run 11.



C. Side view of area marked in A by run 3, two levels of ground can be seen, first from the first pass of the intersection marked in and the second ground is from second pass.



D. Top view of marked area in A by run 3. The double walls can be seen compared to B.

Figure 4.13: The colors are a function of height.

4.4 Test data 3: Large urban data set

Bar graph in Figure 4.14 with the RMSE error compared with reference points and the loop closure error. Table with run settings and more detailed results in Table 4.5. Notable is most method to integrate IMU measurement does visible improvement towards the loop closure error, and the error from using IMU can reduce the loop closure error from the size of 120 to 20, by reducing the angular drift by a factor 5.

Position graph of the run 1 and 11 with reference points compared in Figure 4.15 where a top view and a tilted view of the poses. Cloud generated by run 11 can be found in Figure 4.16, and with more details of location marked A and B in Figure 3.11.

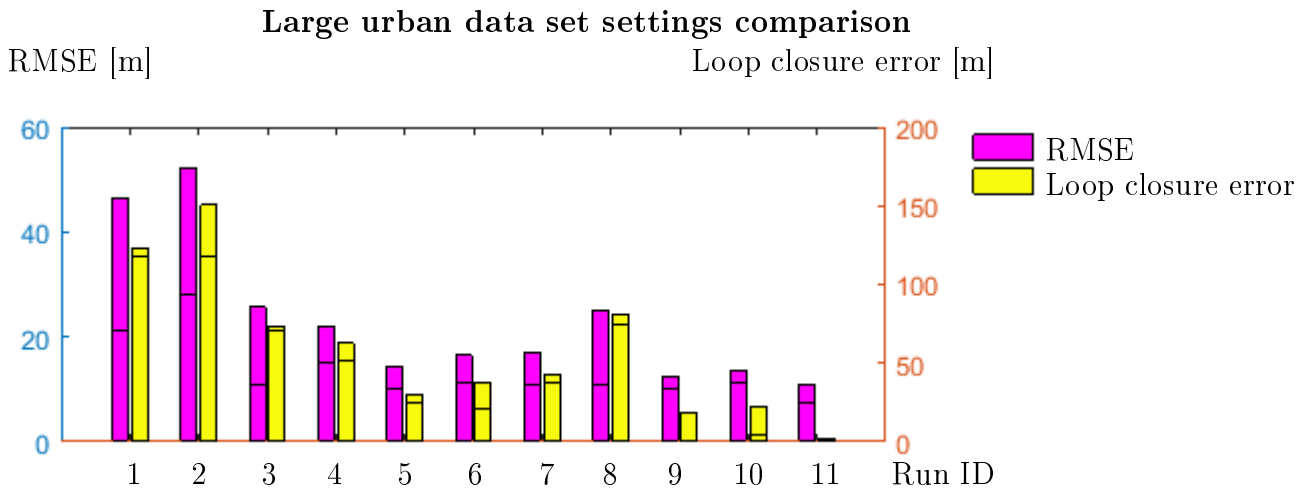
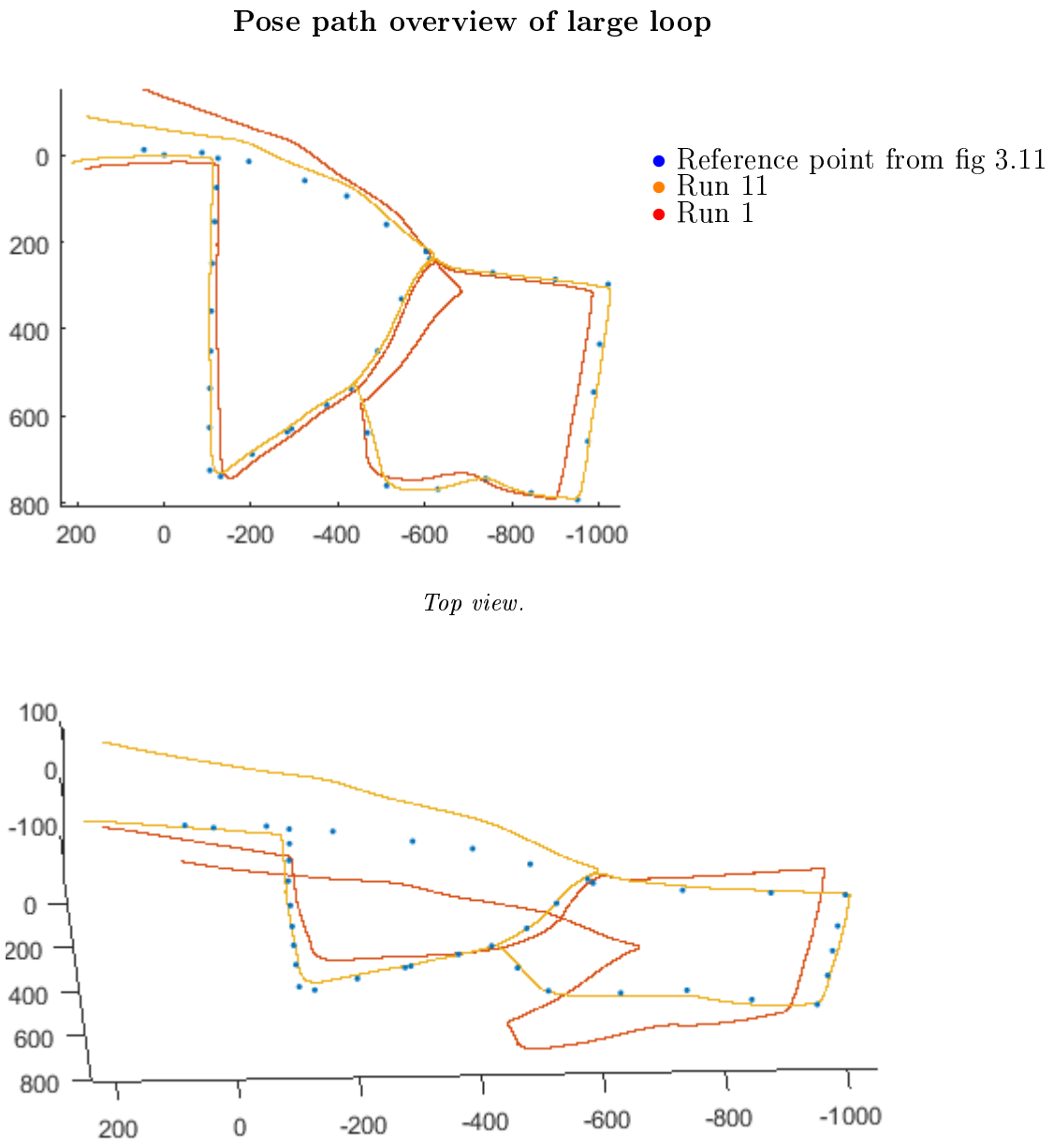


*Figure 4.14: TDash in magenta bar is the RMSE error when projecting pose path onto the XY plane. Dash on yellow bar is the error in Z-axis. * a loop closure error less than 0.2 m. Additional details in Table 4.5.*

Table 4.5: Ends at A/B refers to the location in Figure 3.11 where the process disrupts. The LSO parameter used were $\eta_{freq} = 60, \eta_{dist} = 100$. No loop closure(nlc) marked if the path did not close at location A. Corresponding bar graph in Figure 4.14. LCE is loop closure error.

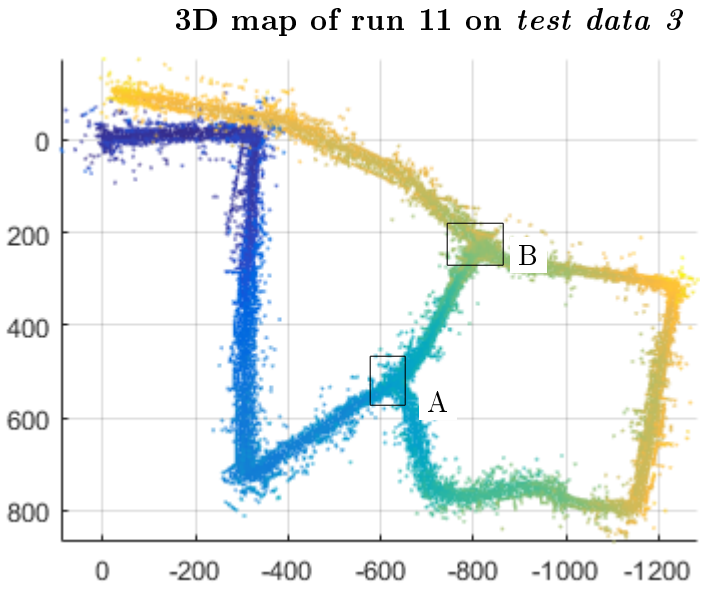
Run	Settings		LSO	mGICP	LCE [m]	LCE [rad]	Drift $10^{-3}\frac{[m]}{m}$	Drift $10^{-5}\frac{[rad]}{m}$	Notes
	IMU	FB	η_{weight}	α_{gicp}					
1	No	P	-	-	∼123	n/a	23	4.6	$\tau = 0$, nlc
2	No	P	-	-	∼150	n/a	n/a	n/a	Ends at A
3	Yes	P	-	-	73.9	0.117	6.3	2.2	Ends at B
4	Yes	V	-	-	62.7	0.209	10.7	1.4	Ends at B
5	Yes	P	0.03	-	30.4	0.094	4.1	0.8	
6	Yes	V	0.03	-	38.0	0.170	4.2	1.0	
7	Yes	P	-	0.3	43.1	0.128	6.3	1.1	Ends at B
8	Yes	V	-	0.3	∼81.3	n/a	14	3	No loop closure
9	Yes	P	0.03	0.3	18.5	0.126	4.8	0.3	
10	Yes	V	0.03	0.3	22.6	0.136	12	0.8	
11	Yes	P	-	-	<0.5	<0.01	-	-	Graph OPT



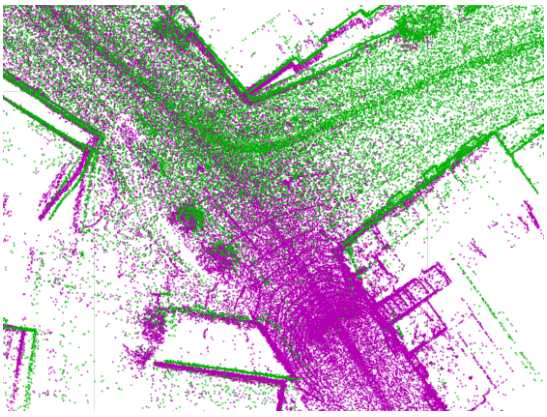
Top view.

Tilted sideview.

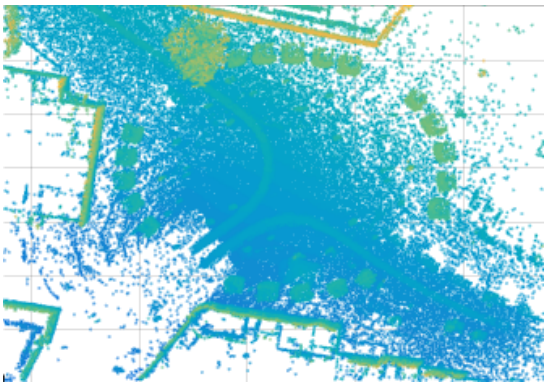
Figure 4.15: Pose path overview of the large loop, by run 1, 11 and reference point from Figure 3.11. All axis unit is given in meter.



C. Cloud density downscaled by factor 10^5.



A. Magnified of loop closed area A in figure A, purple the first pass and green the second pass.



B. Magnified of area B in figure A, cloud down scaled by factor 100.

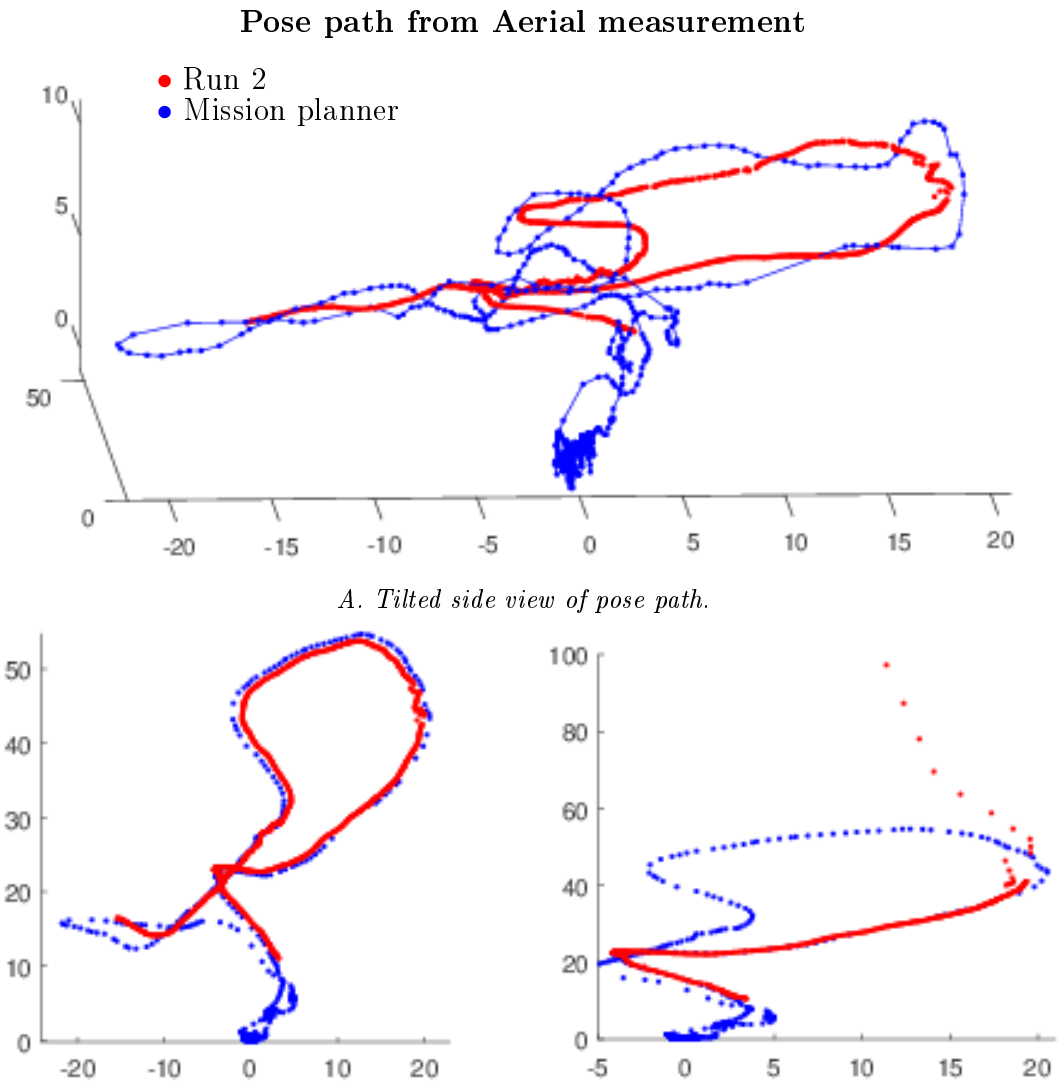
Figure 4.16: 3D map result of the large urban loop with run setting 11. Figure C: Overall run, A: Loop closure area, B: Second intersection.

4.5 Test data 4: Countryside data set

In Table 4.6 we can see a summary of the run applied on the countryside data set. The IMU were limited to 3 Hz and were only used to test the the local sequential optimization algorithm. In Figure 4.17:A we can see the flight path generated by using an EKF filter for the positional feedback and the path by Mission planner software. In Figure 4.17:C we can see the improvement by using an EKF to update the state estimation. A result of the point cloud registration can be seen Figure 4.18.

Table 4.6: Result of ICP-SLAM from aerial view without IMU, RMSE is compared to the mission planner generated pose path, and the RMSE when projected onto XY-plane.

Run	Settings	RMSE	RMSE (Proj)
1	P, $\tau = 0$	incomplete path	incomplete path
2	P, $\tau = 0.001$	1.3321	0.9902
3	P, LSO $\eta_{freq} = 30, \eta_{dist} = 60, \eta_{weight} = 0.03$	1.3164	0.9207
4	P, LSO $\eta_{freq} = 60, \eta_{dist} = 100, \eta_{weight} = 0.03$	1.2665	0.8893



A. Tilted side view of pose path.

B. Flightpath projected onto XY plane.

C. Flightpath by run 1.

Figure 4.17: Comparison of ICP-SLAM with the EKF in mission planner. • path generated by run 2, and • generated by the Mission planner. All axis units are given in meter.

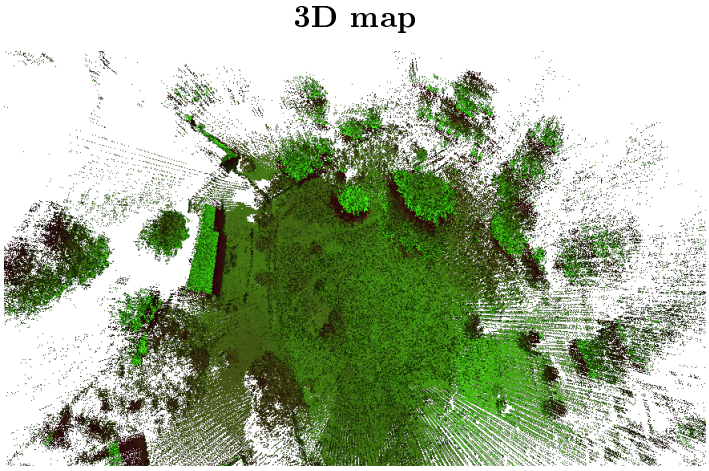


Figure 4.18: 3D map by run 2, color based on height.

Chapter 5

Discussion

This chapter begins with discussion of important parameters and the proposed method utilizing IMU measurement. A discussion of the evaluation method is followed and then the chapter concludes with suggestions for future work.

5.1 Results

5.1.1 Model and observation noise

One of the more important choices to be made is to determine whether or not to use position or velocity feedback and then choose the noise ratio between the model and the observation. In the Table 4.1 we can see a summary of varying noise ratio together with the resulting loop closure error. No notable pattern was discovered, except having $\tau = 0$ causes the forest set to miss align at start. If τ is set too high the solution diverges while for the urban set it were less sensitive of τ. In Tables 4.3, 4.4 and 4.5, we can see the results from varying position and velocity feedback with different settings. Based on the results, neither feedback method is superior to one another as they perform more or less the same.

5.1.2 Frame removal

Table 4.2 and Figure 4.5 shows the result of only using every n:th measurement for the SLAM process. Overall the results indicates that the IMU can improve the robustness if there is a risk of losing measurements, as less measurements are used. The case where the SLAM process fails is when the car goes around the corner. This is an unpredictable behavior which the gyroscope can measure and as shown the gyroscope can improve the robustness.

5.1.3 State prediction with IMU

In Figure 4.2 we can see the effect of using the IMU measurement to update the state estimation for the test data 1 and 2, and and Table 4.5 for test data 3. When using IMU to improve the prediction there seems to be a small improvement when using in combination of position feedback, and no or negative effect in small urban set in combination of velocity

feedback. For the large urban data set the improvement by using position feedback with IMU lead to reducing the loop closure error with 50% compared to without using IMU.

The small effect in the small urban data and forest set, is most likely due to the recording time being relative low and as such any eventual drift has not been accumulated enough where the effect of the IMU can be seen. Even if lower loop closure value would be detected, more data set are still needed to cancel out the improvement being due to random process.

An explanation to the recording time (data set size) would matter could be due to how errors accumulate. An error in estimating the position would only grow linearly and any angular error grows exponentially. The IMU contributes very little to the position estimation, as if we would have an acceleration of $10m/s^2$ then at 10 Hz it would adjust the prediction by 5 cm, and for 20 Hz 1.25 cm. Compared to a motion model without acceleration, which is not a much more improved guess. In the long run the IMU has less drift in angular measurements (roll and pitch can always be estimated) and as such the contribution is more visible in the large urban loop.

5.1.4 Altering the cost function

Surface approximation

Loop closure error when utilizing the alternative surface approximation (Section 2.3) can be seen in Figure 4.2. There is an improvement with the forest data set and mixed results with the small urban data set. In Figure 4.1 we can see the estimated trajectory, where we can see a notable differences in the trajectory between the 1-1 surface and the alternative surface approximation. On the return path we can not that the 1-1 surface approximation occasionally fail to register while the alternative surface does not. This probably reflects that for the forest environment, the existing vertical surfaces are composed of thin vertical surfaces (trees) while the urban environment largely consists of walls.

Assuming a point in a cloud represents a measurement on a tree and we are matching two points from different point cloud to each other. Then it would be more likely that the points are aligned along the vertical axis than the horizontal one. At the return path the majority of the overlap contain measurement from trees (as the LiDAR unit is tilted). Therefore the registration process becomes more sensitive to how the surface is approximated. This could be the reason why the forest set diverges at the end with the 1-1 surface approximation.

Alpha parameter in mGICP

In the Figures 4.3:A,B and D.1:A,B we can see the result of using $mGICP$ with varying size of α_{gicp}. When updating the EKF with cloud registration result as a position, there are possibilities for reducing the loop closure error. When updating the EKF as a velocity the mGICP increases the loop closure error. For the forest data set, a greater interval of α_{gicp} values yielded a smaller loop closure error compared to the small urban dataset.

The reason for mGICP working better with position feedback could be due to the reason stated in Section 2.2.2 where the argument is based on the Figure 2.4. Position feedback absorbs the drift differences between the estimated position and the point cloud

pose, while the velocity feedback does not. The penalty is added relative to difference from estimated position and not difference from estimated travel distance.

Overall improvement in the forest data set could be due to the data set being noisy and contains a lot more false local minima, while adding penalty function increases the confidence of the prediction. One could argue that changing the noise ratio in the EKF would do the same, but changing noise ratio will not affect the existing local minima.

5.1.5 Local Sequence Optimization

In Figure 4.4, we can see the LSO applied with various parameters in combination position feedback and in Figure D.2:A,B in combination of velocity feedback. The result does not indicate that a certain parameter setting is better. If we take into account the measuring frequency, then performing LSO every third second leads to a slight benefit based on these two data sets, but not distinct enough to say it works. Similar nonsignificant results can be seen in Table 4.6 where it was applied into the countryside data set, but lack of effect could be due IMU only recorded at 3 Hz. Overall the result seems mixed, but using the pose of last registration instead of EKF estimate state to fill the unknown states either have no impact or a slight more positive impact.

The reason for improvements being more notable in the forest data set compared to the small urban data set could be due to the LSO primarily perform angular error correction and the measurement contains more angular movement compared to the urban measurement.

5.1.6 Combination of method

For the forest data set (Table 4.3 run 9 and 10) and the large urban data set (Table 4.5 run 9 and 10) an improvement can be seen. The lack of improvement on the small urban data set is probably due to as mentioned earlier, the lap is too short for any angular errors to accumulate big enough for the modification to be notable.

5.2 Evaluation method

5.2.1 Dataset

The lack of distinct results based on the small urban data set compared to the large urban data set could be due to that the data collection was performed in a relative slow speed and in that case the GICP algorithm alone is stable enough to give a good pose estimation.

5.2.2 Evaluation metric

The ground truth comparison with the forest data set does not perform as expected based on the result in Table 4.3. A smaller loop closure error should correlate to a smaller RMSE error when comparing the point clouds towards each other. This is likely due to incorrect associations when comparing the generated 3D map to the ground truth.

The reference trajectory for the urban sets were determined by manually picking out the estimated drive path from a map, which probably lies within 1 meter from the true

position. With an added error from the height estimation of $\pm$ 0.5 m as such any RMSE value below 1.2 is within the noise levels. In that case the RMSE result from small urban data set cannot not be used. The Visual SLAM had a loop closure error of approximately 2 m, then the average error for each point is approximately 1 m. As such any RMSE below approximately 1 is within the noise levels for the forest data set. An ICP algorithm is applied to compare the estimated trajectory with the reference trajectory, as such it is possible that a point in the reference is not the same as the one in the estimated position.

5.3 Further work

By looking into the overall result from the forest data set (Figure 4.6 and the large urban data set (Figure 4.14), majority of the errors for most of the evaluated settings is on the Z-axis. For that reason a barometer is of interest as it could potentially improve a lot of the SLAM process. The roll and pitch estimates from the accelerometer used in the LSO algorithm still relies on an estimate of the yaw angle so they are not completely unbiased of the point cloud registration. This could be dealt with by using a magnetometer which would uncouple the angle estimation from the point cloud registration. If both magnetometer and barometer are added, then it would be possible to optimize after 4 states for the LSO algorithm instead of the current 2 states.

More investigation is also needed on how to interpret the result of GICP and how the value function can be related to the confidence of the match. More investigations are needed to determine how the penalty function should adjust itself depending on the confidence of prediction to motivate the choice of α_{gicp}. To improve the mGICP when using it in combination of velocity feedback, the initial guess should perhaps not be dependent on the current EKF estimated position. It should instead be dependent on the estimated velocity which would ignore the drift mentioned in Section 2.2.2. The EKF could also be improved by setting the gravity as a state as every sensor has an unique bias.

Dynamic point detection could be useful to remove non static objects such as cars in the urban environment which would reduce number of false surfaces. Estimating horizontal surfaces does not work well with the current surface approximation if only one laser can detect it. It could be interesting to investigate if it is better to perform PCA on neighbors in the 2D manifold, instead of just checking the 4 nearest neighbors of the manifold. It would potentially mean that the system is less sensitive if the LiDAR system is rotated along the roll axis.

For evaluation, bringing GPS equipment to determine one's true position with a timestamp would improve over the current method of estimation trajectory. The expected error of estimation the true trajectory would be lower and with timestamps, a correct association can be made between the estimated position and the reference position. This could solve the issue of RMSE being unusable as metric in the small urban data set. A virtual environment would be ideal where the true RMSE can be calculated. Alternatives for the forest data set could be to detect trunk and base the RMSE on trunk location.

More data sample could be evaluated to narrow down what noise ratio to use and to determine in the general case to whether or not to use position feedback or velocity feedback. Then follow up with some parameter optimization algorithm to find better parameter settings for $mGICP$ and the LSO. Data sets with larger loops would help (compared to the small urban data set and forest set) as drift errors can accumulate

more so results does not fall into noise levels. This could also be done in combination of more IMU sources to determine if the quality of an IMU has an effect.

Chapter 6

Conclusion

The aim of this thesis was to create an ICP-SLAM algorithm which can work with sparse LiDAR data, and then investigate if and how IMU data (accelerometer and gyroscope) can be used to improve the SLAM process. The sparse data source tested on were from a Velodyne HDL 32 and a VLP 16 type and from various IMU system.

The proposed method involves using a 6DoF motion model with an EKF as basis for estimation. It takes input processed LiDAR and IMU data to improve the state estimation. The used method for LiDAR data registration is a modified version of the generalized-ICP algorithm. We applied an alternative surface approximation and additional pose graph optimization.

The SLAM evaluations have mainly been based on the error in loop closing. The experimented environments have been in a forest, urban and a countryside environment. The platform used has been a backpack mounted, car mounted, and a hexacopter mounted Velodyne.

The alternative surface approximation in the forest data set has shown to improve the robustness as the resulting trajectory remains reasonable. By utilizing EKF to update the state estimation it has shown to improve the robustness in the forest data set and the countryside data set, by going from a diverging trajectory to a stable one.

The results from the small urban data set and the forest data set have shown similar results with no obvious improvement with different settings. This suggests the overall SLAM framework or the point cloud registration (GICP) is very stable during a short period.

By comparing tests on the small and large urban set results the latter showed results that indicated that the proposed algorithm can perform better than the naive method that only uses GICP for point cloud registration. This is probably due to positional drift accumulates linearly, and error by angular drift accumulates exponentially, where the latter is more notable in larger set. Since the IMU can estimate pitch and roll accurately it would suppress the angular errors more.

Overall more data sets are required and preferably with a longer measuring time such that results are less likely to be within noise levels. It could be used to narrow down parameters settings, especially the observation/model noise ratio and interpreting the GICP results for a general case. Preliminary results show the proposed model does provide reasonable trajectories with indication that improvement can be done by utilizing IMU data in an ICP-SLAM framework.

Bibliography

[1] Andreas A. Nuchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6d slam 3d mapping outdoor environments. *Journal of Field Robotics*, 24(8-9):699–722, 2007.

[2] A. Aditya. Implementation of a 4d fast slam including volumetric sum of the uav. In *Sensing Technology (ICST), 2012 Sixth International Conference on*, pages 78–84. Institute of Electrical & Electronics Engineers (IEEE), Dec 2012.

[3] M Alpen, C Willrodt, K Frick, and J Horn. On-board slam for indoor uav using a laser range finder. In *SPIE Defense, Security, and Sensing*, pages 769213–769213. International Society for Optics and Photonics, 2010.

[4] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[5] C. G. BROYDEN. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

[6] M. Bryson and Salah Sukkarieh. Active airborne localisation and exploration in unknown environments using inertial slam. In *Aerospace Conference, 2006 IEEE*, pages 13 pp.–, 2006.

[7] F. Caballero, L. Merino, J. Ferruz, and A. Ollero. Vision-based odometry and slam for medium and high altitude flying uavs. *Journal of Intelligent and Robotic Systems*, 54(1):137–161, 2008.

[8] A. Censi. An icp variant using a point-to-line metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 19–25, May 2008.

[9] G. Conte G. Scaradozzi D. Zanoli S. M. Gambella L. & Marani. Underwater slam with icp localization and neural network objects classification. *International Society of Offshore and Polar Engineers.*, 2008.

[10] D.W. Eggert, A. Lorusso, and R.B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9(5):272–290.

[11] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572, 1901.

[12] Christina Gronwall, Gustaf Hendeby, and Kristian Sinivaara. A proposal for combining mapping, localization and target recognition. In *SPIE Security+ Defence*, pages 96490G–96490G. International Society for Optics and Photonics, 2015.

[13] Kyuseo Han, C. Aeschliman, J. Park, A. C. Kak, Hyukseong Kwon, and D. J. Pack. Uav vision: Feature based accurate ground target localization through propagated initializations and interframe homographies. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 944–950, May 2012.

[14] D. Holz and S. Behnke. Sancta simplicitas - on the efficiency and achievable results of slam using icp-based incremental registration. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1380–1387, May 2010.

[15] Dirk Holz and Sven Behnke. Registration of non-uniform density 3d laser scans for mapping with micro aerial vehicles. *Robotics and Autonomous Systems*, 74, Part B:318 – 330, 2015. Intelligent Autonomous Systems (IAS-13).

[16] Yani Ioannou, Babak Taati, Robin Harrap, and Michael Greenspan. Difference of normals as a multi-scale operator in unorganized point clouds. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 501–508. IEEE, 2012.

[17] P. Jensfelt, D. Kragic, J. Folkesson, and M. Bjorkman. A framework for vision based bearing only 3d slam. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1944–1950, May 2006.

[18] Erika Bilock Joakim Rydell. Panther chameleon: Real-time image-based positioning and mapping. *Proceedings of the 2015 International Technical Meeting of The Institute of Navigation*, pages 768 – 777, January 2015.

[19] Lantmäteriet. `https://kso.etjanster.lantmateriet.se/`. [Online; accessed 29-June-2016].

[20] Lantmäteriet. `https://www.lantmateriet.se/sv/Kartor-och-geografisk-information/Hojddata/GSD-Hojddata-grid-2/`. [Online; accessed 29-June-2016].

[21] T. Lemaire and S. Lacroix. Monocular-vision based slam using line segments. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2791–2796, April 2007.

[22] John J Leonard and Hans Jacob S Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *ROBOTICS RESEARCH-INTERNATIONAL SYMPOSIUM-*, volume 9, pages 169–178. Citeseer, 2000.

[23] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275.

[24] J. W. Marck, A. Mohamoud, E. v. d. Houwen, and R. van Heijster. Indoor radar slam a radar application for vision and gps denied environments. In *Radar Conference (EuRAD), 2013 European*, pages 471–474, Oct 2013.

[25] F. Moosmann and C. Stiller. Velodyne slam. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 393–398, June 2011.

[26] A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d slam with approximate data association. In *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, pages 242–249, July 2005.

[27] A. Nuchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 6d slam with an application in autonomous mine mapping. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1998–2003 Vol.2, April 2004.

[28] AE Oguz and Hakan Temeltas. On the consistency analysis of a-slam for uav navigation. In *SPIE Defense+ Security*, pages 90840R–90840R. International Society for Optics and Photonics, 2014.

[29] A. Palomer, P. Ridao, D. Ribas, A. Mallios, N. Gracias, and G. Vallicrosa. Bathymetry-based slam with difference of normals point-cloud subsampling and probabilistic icp registration. In *OCEANS - Bergen, 2013 MTS/IEEE*, pages 1–8, June 2013.

[30] Michael JD Powell. A fortran subroutine for solving systems of nonlinear algebraic equations. Technical report, Atomic Energy Research Establishment, Harwell (England), 1968.

[31] D. Rao, S. J. Chung, and S. Hutchinson. Curveslam: An approach for vision-based navigation without point features. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4198–4204, Oct 2012.

[32] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: Science and Systems*, volume 2, 2009.

[33] Gerald L Smith, Stanley F Schmidt, and Leonard A McGee. *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. National Aeronautics and Space Administration, 1962.

[34] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[35] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.

[36] R. Tiar, M. Lakrouf, and O. Azouaoui. Fast icp-slam for a bi-steerable mobile robot in large environments. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 611–616, July 2015.

[37] M. Tomono. Robust 3d slam with a stereo camera based on an edge-point icp algorithm. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4306–4311, May 2009.

[38] Velodyne. `http://velodynelidar.com/docs/manuals/`. [Online; accessed 20-August-2016].

[39] R. Wagner, U. Frese, and B. Bauml. Graph slam with signed distance function maps on a humanoid robot. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2691–2698, Sept 2014.

[40] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1457–1464, 2013.

Appendix A

Parameter values and notations

Category	Sign	Default value	Description
EKF noise	ϵ_{acc}	0.05^2	Acceleration model noise
	ϵ_{angvel}	1^2	Angular velocity model noise
	σ_{acc}	0.08^2	IMU accelerometer observation noise
	σ_{angvel}	0.08^2	IMU gyroscope observation noise
	τ_{pos}	0.01^2	GICP position observation noise
	τ_{ang}	0.01^2	GICP angle observation noise
GICP	γ_{dr}	4	Default number of recent frames to compare against
	γ_{dk}	2	Default number of key frames to compare against
	γ_{kr}	6	Number of recent frames to compare with for key-frames
	γ_{kk}	6	Number of key-frames to compare with for key-frames
	$\gamma_{trans\ conv}$	0.005	Convergence criteria for translation
	$\gamma_{rot\ onv}$	0.001 rad	Convergence criteria for rotation
	$\gamma_{nn\ search}$	1.5	Nearest neighbor search distance
mGICP	α_{gicp}	0.3	Weight on added penalty function
		0.05 m	Minimal stdev on position
		2.5°	Minimal stdev on angle
BFGS		20	Number of outer iteration
		10	Number of inner iteration
Mesh size		5×3	Mesh size for 32 scan lines
		3×2	Mesh size for 16 scan lines
LSO	η_{freq}	30 frames	How frequent to LSO is applied (10 Hz scan)
	η_{freq}	60 frames	How frequent to LSO is applied (20 Hz scan)
	η_{Range}	60 frames	How far back LSO optimizes edges (10 Hz scan)
	η_{Range}	100 frames	How far back LSO optimizes edges (20 Hz scan)
	$\eta_{angstab}$	0.2 rad	Limit of movement if LSO is applied
	η_{weight}	0.03	Weight on the filled in state in LSO
Misc.	Surface type		Proportional size

Appendix B

Coordinate transform

The intrinsic coordinate system axis are determined by rotating the coordinate system in extrinsic in the order of $R_z R_y R_x$ where R_i is a rotation matrix which rotates clockwise around the extrinsic axis i. If the angular pose is pry (pitch roll yaw) around the respective axis x-y-z, then each rotation matrix is

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(p) & -\sin(p) \\ 0 & \sin(p) & \cos(p) \end{bmatrix} \quad R_y = \begin{bmatrix} \cos(r) & 0 & \sin(r) \\ 0 & 1 & 0 \\ -\sin(r) & 0 & \cos(r) \end{bmatrix} \quad R_z = \begin{bmatrix} \cos(y) & -sin(y) & 0 \\ \sin(y) & cos(y) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.1}$$

and the complete rotation matrix

$$R = R_z R_y R_x = \tag{B.2}$$

$$\begin{bmatrix} \cos(r)\cos(y) & \cos(y)\sin(p)\sin(r) - \cos(p)sin(y) & \sin(p)\sin(y) + \cos(p)\cos(y)\sin(r) \\ \cos(r)\sin(y) & \cos(p)\cos(y) + \sin(p)\sin(r)\sin(y) & \cos(p)\sin(r)\sin(y) - \cos(y)\sin(p) \\ -\sin(r) & \cos(r)\sin(p) & \cos(p)\cos(r) \end{bmatrix} \tag{B.3}$$

Differentiation of rotation matrix

Rotation matrix derivative with respect to pitch angle p

$$\frac{\partial R}{\partial p} = \begin{bmatrix} 0 & \sin(p)\sin(y) + \cos(p)\cos(y)\sin(r) & \cos(p)\sin(y) - \cos(y)\sin(p)\sin(r) \\ 0 & \cos(p)\sin(r)\sin(y) - \cos(y)\sin(p) & -\cos(p)\cos(y) - \sin(p)\sin(r)\sin(y) \\ 0 & \cos(p)\cos(r) & -\cos(r)\sin(p) \end{bmatrix} \tag{B.4}$$

derivative with respect to roll angle r

$$\frac{\partial R}{\partial r} = \begin{bmatrix} -\cos(y)\sin(r) & \cos(r)\cos(y)\sin(p) & \cos(p)\cos(r)\cos(y) \\ -\sin(r)\sin(y) & \cos(r)\sin(p)\sin(y) & \cos(p)\cos(r)\sin(y) \\ -cos(r) & -\sin(p)\sin(r) & -\cos(p)\sin(r) \end{bmatrix} \tag{B.5}$$

derivative with respect to yaw angle y

$$\frac{\partial R}{\partial y} = \tag{B.6}$$

$$\begin{bmatrix} -\cos(r)\sin(y) & -\cos(p)\cos(y) - \sin(p)\sin(r)\sin(y) & \cos(y)\sin(p) - \cos(p)\sin(r)\sin(y) \\ \cos(r)\cos(y) & \cos(y)\sin(p)\sin(r) - \cos(p)\sin(y) & \sin(p)\sin(y) + \cos(p)\cos(y)\sin(r) \\ 0 & 0 & 0 \end{bmatrix} \tag{B.7}$$

Appendix C

Complementary filter

Complementary filter update formula for estimating roll and pitch for the LSO.

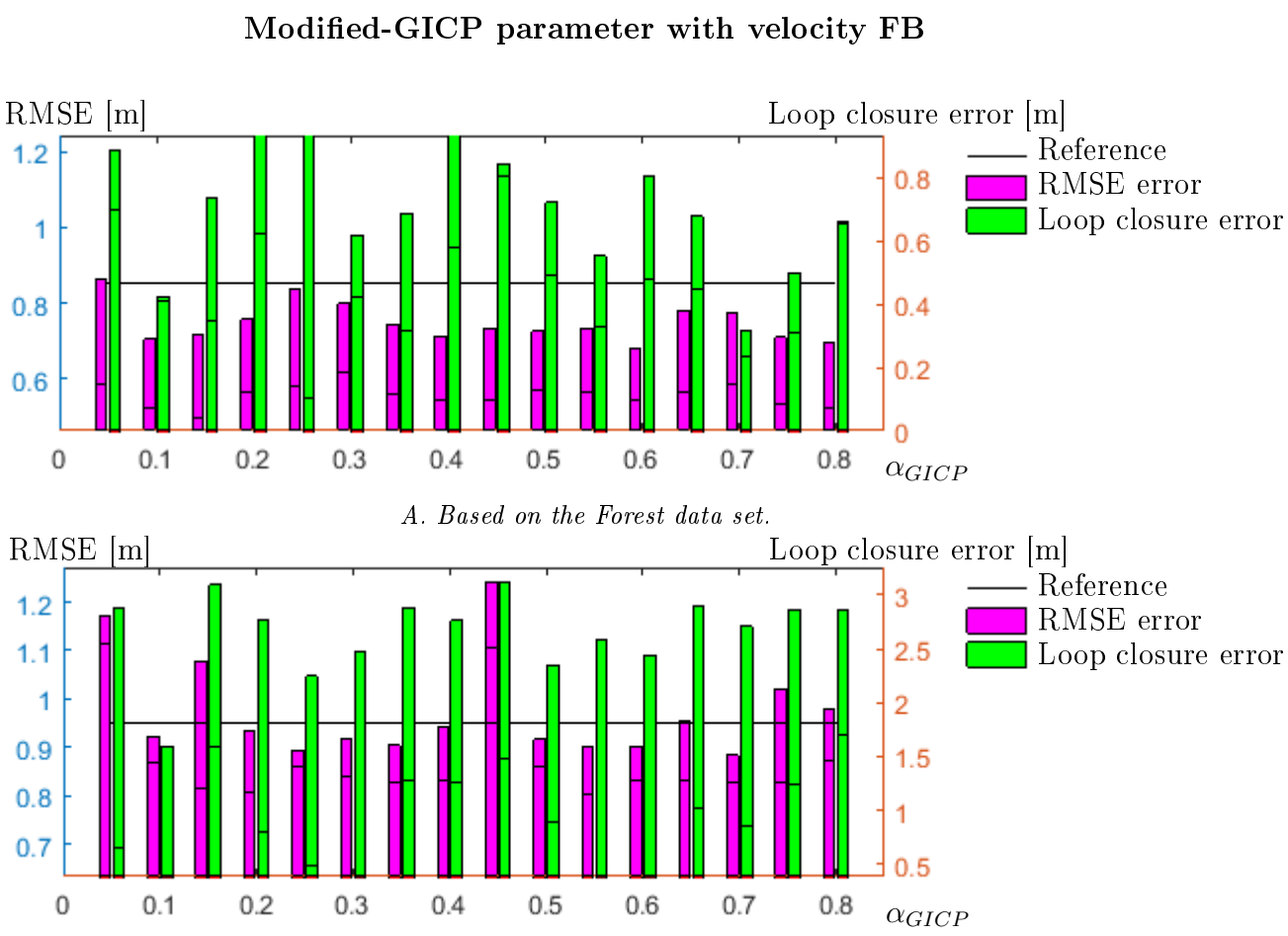
$$\begin{bmatrix} \text{Pitch}(t+h) \\ \text{Roll}(t+h) \\ \sim \end{bmatrix} = 0.98 \left(\begin{bmatrix} \text{Pitch}(t) \\ \text{Roll}(t) \\ \sim \end{bmatrix} + h \cdot R_z(y) R_y(r) R_x(p) \begin{bmatrix} \dot{p} \\ \dot{r} \\ \dot{y} \end{bmatrix} \right) + \tag{C.1}$$

$$0.02 R_z \begin{bmatrix} \arctan(\frac{a_y}{a_z}) \\ \arctan(\frac{-a_x}{\sqrt{a_y^2 + a_z^2}}) \\ \sim \end{bmatrix} \tag{C.2}$$

Appendix D

Misc Results

Additional results from varying α_{gicp} (Figure D.1) and LSO parameter (Figure D.2) combination of velocity feedback.



A. Based on the Forest data set.

B. Based on the small urban data set.

Figure D.1: The α parameter in the modified-GICP cost function has been varied from 0.05 to 0.8 with 0.05 intervals, in combination of Velocity feedback. The reference bar is based on run 4 (Velocity feedback with IMU). Green area under the mark for each bar is the error located in Z-axis, and magenta area under the mark is RMSE when projecting pose path onto XY plane.

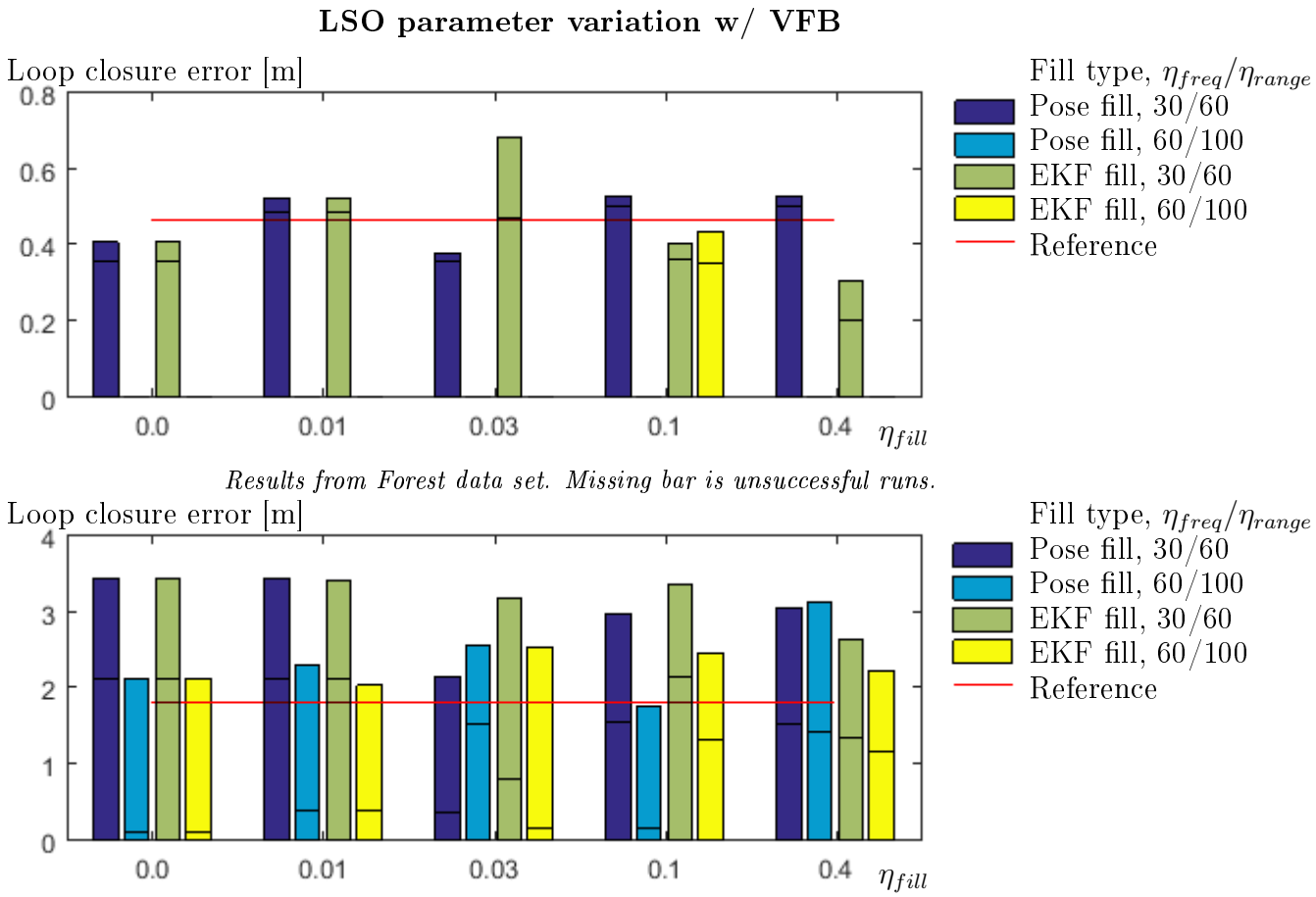


Figure D.2: LSO parameter variation applied onto forest and small urban data set in combiantion of velocity feedback. Reference is based on run 4 (Velocity feedback with IMU). Area under the black bar is the error located on the Z-axis.

TRITA -MAT-E 2016:66
ISRN -KTH/MAT/E--16/66--SE