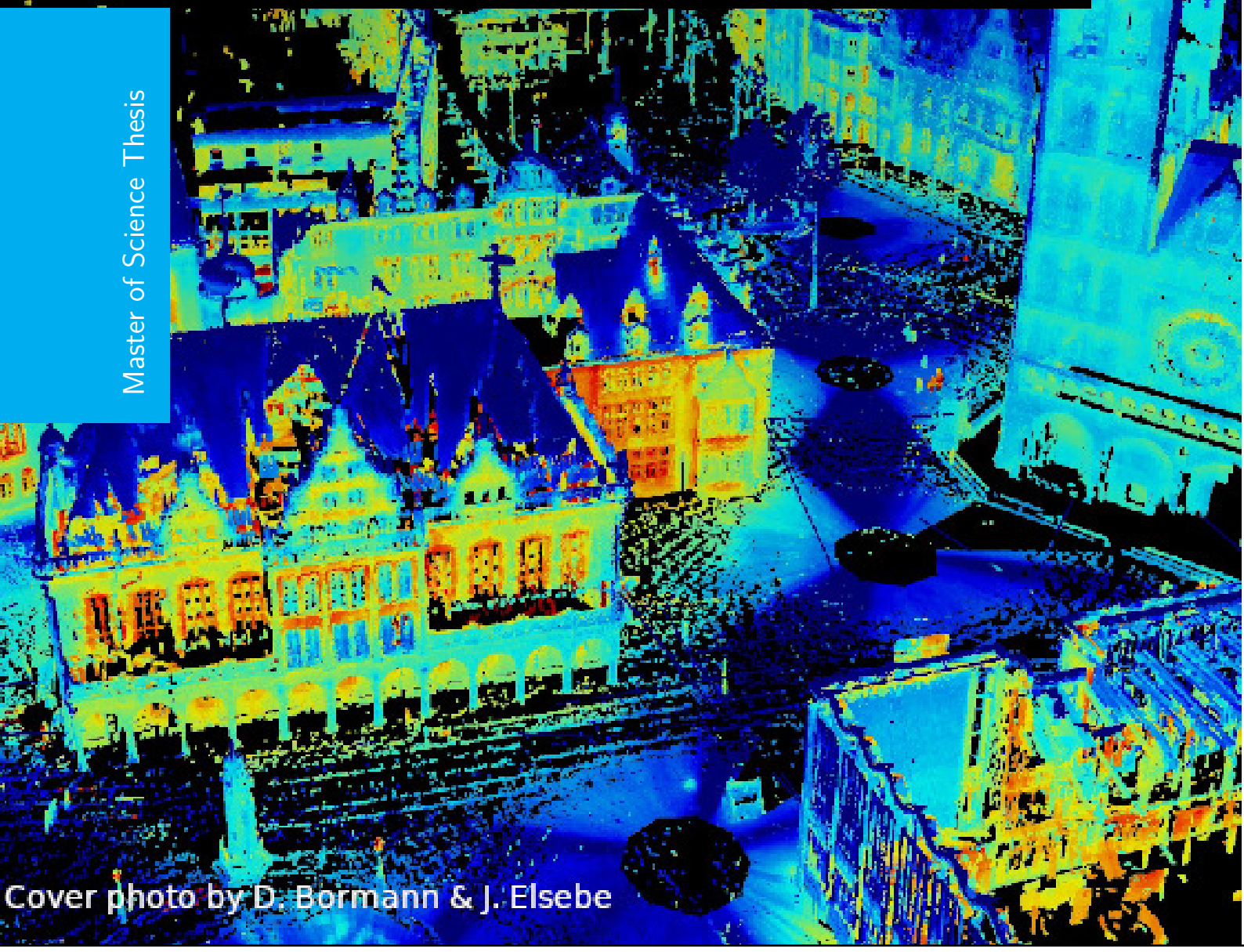
Graph Based LiDAR-Inertial Localization with a Low Power Solid State LiDAR

AF Vonk

Master of Science Thesis



Cover photo by D. Bormann & J. Elsebe

TU Delft Delft University of Technology

Delft Center for Systems and Control

Graph Based LiDAR-Inertial Localization with a Low Power Solid State LiDAR

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

AF Vonk

March 28, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was carried out in cooperation with CGI The Netherlands.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Graph Based LiDAR-Inertial Localization with a Low Power Solid State LiDAR

by

AF Vonk

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: March 28, 2022

Supervisor(s):

dr.ir. C.S.Smith

ir. R. Voûte

Reader(s):

dr.ir. A.J.J. van den Boom

Abstract

Mapping an environment with a Light Detection and Ranging (LiDAR) sensor through the use of a LiDAR Simultaneous Localization And Mapping (SLAM) algorithm is a powerful technology that allows for the creation of detailed 3D models. Recently various LiDAR sensors have been developed based on Micro-Electro-Mechanical System (MEMS) technology. These LiDARs are very low cost and considerably smaller than conventional LiDARs. They also often incorporate other sensors such as Inertial Measurement Unit (IMU)s and cameras into the same device.
Performing LiDAR SLAM with MEMS based LiDAR is challenging due to the short range, the smaller Field of View (FOV) and the sensitivity to ambient light of MEMS based LiDAR. In this thesis the objective is to reduce the effect of these factors when doing LiDAR SLAM by incorporating IMU measurements into the position estimation of the sensor.
A graph based positioning approach is proposed to achieve tight coupling of the IMU sensor and LiDAR position estimates. The method is made more robust by incorporating an outlier detection mechanism that reduces the influence of wrong LiDAR position estimates caused by insufficient points in the LiDAR FOV or by ambient light disturbance.
The method was built in ROS and implemented on the Intel® L515 sensor. The performance is evaluated in indoor situations with varying presence of ambient sunlight and where room size approaches the maximum limit of the sensor range. The algorithm achieves lower drift than the current state of the art for the Intel® L515. The algorithm especially achieves altitude drift reduction and increases robustness to outliers in the LiDAR positioning.

Table of Contents

List of Figures

List of Tables

Preface & Acknowledgements

Being stuck in a room while feeling lost is unfortunately the experience a lot of master students had when they were working on their thesis during the corona pandemic.
How nice it would have been if you had a device that you could put on your head to map the environment and find a way out of that situation.
Jokes aside. I really enjoyed working in the field of Simultaneous Localization and Mapping. It felt like the further I got into the thesis, and the more skills I developed handling the Intel L515, the more I was able create my own map out of the room I was stuck in. (re)Discovering points of interest, into a better, more interesting, and more enjoyable life. Since childhood I've always been obsessed with creating (virtual) machines. I love solving the spatial puzzle (in a wide sense) of which piece needs to be placed where, when, and in what order, to create something that does something new or in a different way.
The pieces I used in my life were first bricks, later self-made parts, and after a quick detour of play and screenplay lines: lines of code.
Control allows the creation of the ultimate machine: one that functions on its own.
Since Robert Voûte introduced me to the Intel Sensor and we created the topic, I've been intrigued by what is already possible in the field of SLAM, and very excited about the possibilities that the future will hold.

I would like to thank my supervisor dr.ir. C.S.Smith for his assistance, insight, guidance, honesty and patience during the writing of this thesis.
I would also like to thank Robert Voûte for making me feel at home within CGI.
Furthermore I would like to thank Annet, Jan, Annet, Jeroen, Eline, Dorinde, Laura, David, Jasper, Irek, Laurens, Jim, Johan, Mark, Mels, Arend, Max, Kitty, Jan, Sara, Joris, Trijntje, Muus, Annie, Sjoerd, de Neven, Jimi, Douglas, Cal, Gary, Viktor, Mihalyi, and Martin for giving me strength, insight and love during this thesis.

I hope that by reading this thesis you also, like me, become fascinated by the field of Simultaneous Localization and Mapping. If not, believe me, I had worse audiences.

Delft, University of Technology — AF Vonk
March 28, 2022

"Contrary to what we usually believe, moments like these, the best moments in our lives, are not the passive, receptive, relaxing times—although such experiences can also be enjoyable, if we have worked hard to attain them. The best moments usually occur when a person's body or mind is stretched to its limits in a voluntary effort to accomplish something difficult and worthwhile. Optimal experience is thus something that we make happen. For a child, it could be placing with trembling fingers the last block on a tower she has built, higher than any she has built so far; for a swimmer, it could be trying to beat his own record; for a violinist, mastering an intricate musical passage. For each person there are thousands of opportunities, challenges to expand ourselves."

— *Mihaly Csikszentmihalyi*

Chapter 1

Introduction

The Greek philosopher Anaximander (610–546 BC) has been credited with having created one of the first maps of the world [8]. The map contained 3 continents: Libya (Africa), Europe and Asia. Split by the river Nile, the Mediterranean sea and the Black sea.
Maps nowadays are more often generated by satellites instead of ancient Greek philosophers. Satellite images can reach a resolution of up to 15 cm [3], providing very detailed pictures of environments, and a good start for creating a map. Many things can be localized on those maps by using other satellites that emit GNSS signals and a GPS.
If one wants to monitor aeroplanes, trains or cars, they can also be placed on the map by outfitting them with a GPS.
The maps and monitoring systems created in this way have allowed humankind to create safer flying conditions, more efficient railway systems and up to date traffic information.
But as great as these systems are, due to their reliance on GNSS, they break down when localizing and mapping in indoor environments.

The current state of the art in indoor localization and mapping is reliant on Simultaneous Localization And Mapping (SLAM) techniques. SLAM tries to solve the chicken and egg problem of creating a map of an environment and positioning oneself in that map.
The map and the motion model of the agent in the map can be 2D, but also 3D when one wants to capture additional information of the complexities of the indoor structure.
SLAM has been a breakthrough technology in various field where autonomous operation in unknown environments is needed, or where a map or model of an environment is wanted. Example use-cases include autonomous driving, building modelling, unmanned autonomous flight and underground localization.
Currently one of the most detailed ways to build an indoor 3D model of a building is by using a Light Detection and Ranging (LiDAR) sensor and performing LiDAR SLAM. The laser pulses emitted by a LiDAR provide very accurate distance measurements, allowing for detailed maps.
LiDAR technology is however expensive and the equipment used is also heavy. Especially compared to doing visual SLAM with a camera.
The prohibitive cost and size of LiDAR is starting to change however in recent years since

Micro-Electro-Mechanical System (MEMS) technology has been introduced in LiDAR scanning. MEMS technology has in the past drastically reduced the cost and size of a variety of sensors by scaling down the mechanical parts and using integrated circuit batch processing techniques.
One of the first generation of MEMS based LiDAR devices, the Intel® RealSense™ LiDAR Camera L515 (Intel L515), has been introduced to the market in 2020. This device is designed for indoor use, is very small, and costs a fraction of what a conventional mechanical non-MEMS based lidar costs.
It does however suffer from poor performance when ambient sunlight is present due to its lower power, and its smaller Field of View (FOV) causes degeneracy problems with current LiDAR SLAM methods.
The Intel L515 does however have the advantage of also accommodating a camera and Inertial Measurement Unit (IMU) into the device, that can also be used for SLAM. In this thesis, the advantage of fusing IMU data with LiDAR data for to the accuracy and robustness of LiDAR SLAM with the Intel L515 is researched.
In this chapter the SLAM problem is introduced, and the current state of the art in LiDAR slam with or without other sensors is reviewed.
Then the Intel L515 is introduced and compared to conventional LiDARs.
The problem statement is then introduced which is the guideline for the rest of the thesis.

1-1 Simultaneous Localization And Mapping

When doing SLAM, one tries to create a map of an environment while also positioning oneself in that environment.
SLAM was first mentioned in 1995 [4] and has been a widely researched topic ever since. The SLAM problem can be described as a Maximum A Posteriori (MAP) problem (1-1).

$$P(M, x_{1:t}|y_{1:t}, u_{1:t}) \tag{1-1}$$

Here M is the map that is created and $\mathbf{x}_{1:t}$ is the state of the observer in the environment at various points in time.
The states are estimated with sensor observations $y_{1:t}$, as well as observed inputs to the system $u_{1:t}$.
The sensor observations can come from a variety of sensors. There are sensors that measure the location or orientation of the observer directly like GNSS or magnetometers, but sensors that don't measure the state directly are also commonly used like for example (stereo)camera's, accelerometers, LiDARs, sonars, radars etc.
Possible solutions for the SLAM problem vary and are highly dependent on which sensor is used.
In this thesis a 3 dimensional environment is assumed, and hence the map is 3 dimensional, and the position and orientation of the sensor both have 3 dimensions.

1-1-1 LiDAR based SLAM

A LiDAR sensor measures the distance to points in its FOV. The points are measured at a set interval angle that ultimately covers the FOV. These points are often visualized in three dimensional space as a point cloud.
When the LiDAR moves through the environment, new points are captured as they enter the FOV of the LiDAR. Previously captured points are also recaptured but will be at a different location with respect to the LiDAR because of the movement of the sensor.
These previously captured points are the key to finding the movement of the LiDAR.
LiDAR SLAM methods estimate the movement of the sensor by quantifying the movements of points that have been observed. LiDAR SLAM methods roughly preform these two steps:

1. Points observed in a new scan are matched to the same point observed in a previous scan (point matching step).
2. The motion of the LiDAR sensor is estimated based on the distance between these point matches (motion estimation step).

The Iterative Closest Points algorithm

One of the first methods that does these steps is the Iterative Closest Points (ICP) algorithm [1]. In ICP, the newly observed point cloud is compared to the previously observed point cloud. ICP roughly walks through these four steps in order:

1. ICP does point matching for every point in the newly observed point cloud by finding the closest point in the previously observed point cloud.
2. The motion is estimated by performing a least squares optimization that minimizes the distance between matches as a function of the LiDAR motion.
3. The newly observed point cloud is then transformed with this motion.
4. ICP returns to step 1. After a number of iterations the final relative movement is obtained.

ICP is generally too slow for real time performance because of the large number of points that a LiDAR gathers. The LiDAR point cloud can be sampled down, but that is not desirable, since then useful information is lost.
Because ICP matches the closest points with each other, it does not necessarily matches points that are at the same location in the world. These false matches are quite common with ICP and make ICP prone to end up in a local minima for the relative motion in the optimization step. A good initial guess can reduce the chance of ending in a local minima, but that may not always be available.
The chance of ending in a local minima with ICP is even higher in 3D because of the extra dimensions adding variables and options for false point matches.

The Lidar Odometry and Mapping algorithm and Extensions

The alternatives to ICP that perform the best nowadays are heavily inspired by the LiDAR Odometry And Mapping (LOAM) [22] algorithm.
LOAM is able to achieve real time performance by making some key differences compared to ICP. First LOAM performs a feature detection step before matching points. The feature detection step detects plane and edge points, which allows for a better description of the properties of the newly observed pointcloud.
Because the plane and edge points correlate to a line in the case of an edge, or a plane in the case of a plane, the point-to-line distance or point-to-plane distance can be used as a description during the motion estimation step. The workflow of LOAM is as follows:

1. Edge and plane points are extracted from the unordered point cloud produced by the LiDAR.
2. For these edge and plane points the closest two or three previously observed correspondences are found (two for edges, and three for planes).
3. LOAM uses these previously observed correspondences to describe edge lines or planes.
4. The motion is obtained by minimizing the distance of the newly observed points to those edges and planes by varying the pose of the LiDAR.

After the position of the sensor is determined, the newly observed points are projected to the global map model of LOAM. The result of a point cloud model created with LOAM is

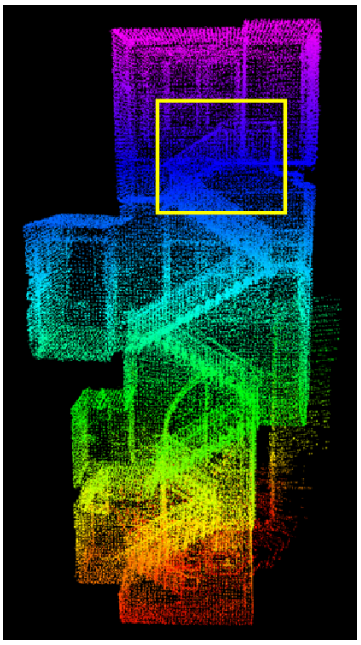
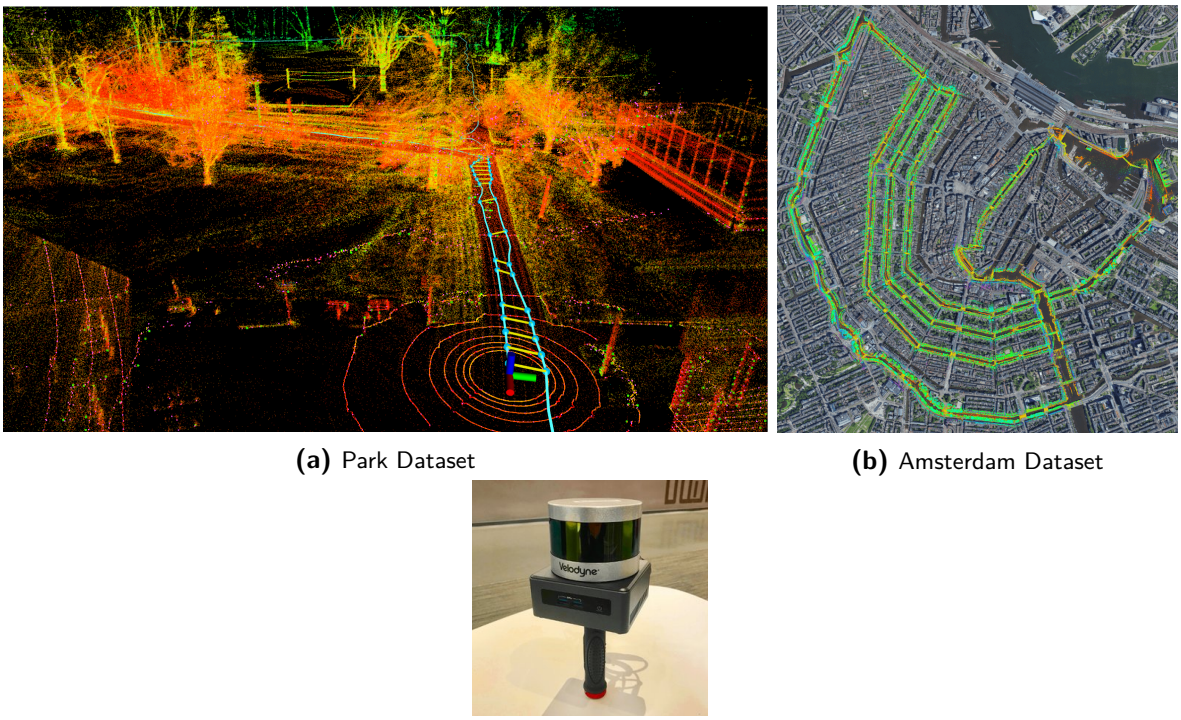


Figure 1-1: Point Cloud of a staircase created with the Lidar Odometry and Mapping (LOAM) algorithm [22].

presented in figure 1-1. Over the years LOAM [22] has been improved with various extensions. LeGO - LOAM [16] extends LOAM by splitting the pose estimation in two separate procedures. First the ground plane is extracted which is used to estimate the yaw, pitch and height of the pose. Then the other feature points are used to estimate the rest of the pose.
IMU's have also been used in combination with LOAM algorithms. The fusion of the IMU in these Lidar Inertial algorithms can be distinguished in two classes: Loosely coupled and tightly coupled.
In loosely coupled algorithms the IMU position estimates are used as a prior for the optimization done by the LiDAR algorithm. In tightly coupled algorithms, the final localization estimates are a direct combination of IMU based estimates and LiDAR based estimates.
Loose coupling of the IMU was already done by the original LOAM algorithm. Where the IMU could be used as an initial guess for the optimization.
Tight coupling of the IMU can be achieved through various Kalman filter approaches [23, 16, 21] or by using a graph based approach [17] [18].
The advantage of using a graph based approach over a filtering based approach is that it allows for the implementation of loop closures.
Loop closures are movement estimates between states that are close to each other in the spatial domain, but can greatly differ in time. Loop closures greatly reduce long term drift [17].
The Lidar Inertial Odometry via Smoothing And Mapping (LIO-SAM) algorithm uses a graph based approach to combine LiDAR estimates, IMU estimates, and (optionally) GPS measurements. Loop closures are created by performing scan matching between non sequential scans that are estimated to be within a certain euclidean distance of each other. The results of LIO-SAM are shown in Fig. 1-2. The loop closures are the yellow matches in (Fig 1-2a). These are created in a process on a separate thread, and added to the factor graph



(a) Park Dataset

(b) Amsterdam Dataset

(c) Handheld setup

Figure 1-2: (a,b) Overview of the LIO-SAM algorithm results (c) The LIO-SAM setup [17].

when found. LIO-SAM is able to maintain positioning over long intervals as can be seen by the Amsterdam dataset performance (Fig. 1-2b). The LIO-SAM datasets were created wit a Velodyne VLP-16 Mechanical LiDAR and a MicroStrain 3DM-GX5-25 IMU. In the Amsterdam dataset GPS measurements were also used in the optimization.

The approaches discussed up until this point have only been tested with expensive mechanical LiDARs. Literature for MEMS based LiDARs is scarce, because of their recent introduction. The first and only (at the time of writing) LiDAR SLAM algorithm for the Intel L515 is the Solid State Lidar Simultaneous Localization and Mapping (SSL-SLAM) algorithm.
SSL-SLAM [20] takes a similar approach as LOAM with a different metric for feature detection. This algorithm provides functional SLAM inside a laboratory environment. SSL-SLAM loses track in high ambient light or open rooms because the LiDAR of the Intel L515 is not able to obtain (usable) feature points in that case. This causes the algorithm to lose track in rooms where there are open areas as well as when the sensor is confronted by sunlight coming from windows. These problems are further elaborated in Sec. 1-2-2.

1-2 Sensor Overview

The Intel L515 consists of a LiDAR sensor, an IMU and a camera (Fig. 1-3). The lidar sensor is a MEMS based scanning LiDAR. The intertial measurement unit is a Bosch BMI085 [10]. In Sec. 1-2-1 the specifics of the Intel sensor are briefly introduced. In Sec. 1-2-2, the challenges and differences between the Intel L515 and conventional LiDAR is explained. The IMU sensor is briefly reviewed in Sec. 1-2-3 to finalize the section.

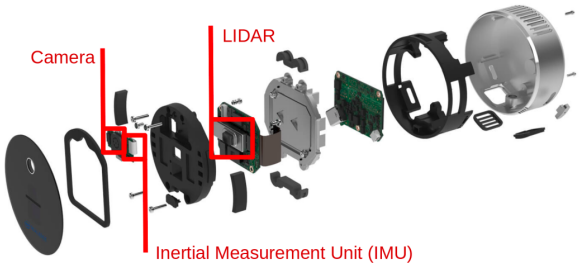


Figure 1-3: Expanded view of all the components of the Intel L515 Sensor. The sensors contains a camera, Lidar and Inertial measurement unit [10].

1-2-1 Lidar Methods

A LiDAR measures distance to surfaces in its environment. A LiDAR emits laser pulses to measure distance. When these laser pulses return to the sensor after reflection, the time of flight is used to determine distance.
To measure space and create a point cloud, the laser needs to cover an area. The area covered by the laser beam is called the FOV of the LiDAR.

LiDAR sensors can be divided in two groups based on how they cover their FOV [19]: scanning LiDAR and non-scanning LiDAR.
The scanning LiDARs can be divided in three subgroups [19]: Optical Phase Array Scanning LiDAR, Mechanical scanning LiDAR and MEMS based scanning LiDAR (Fig.1-4)

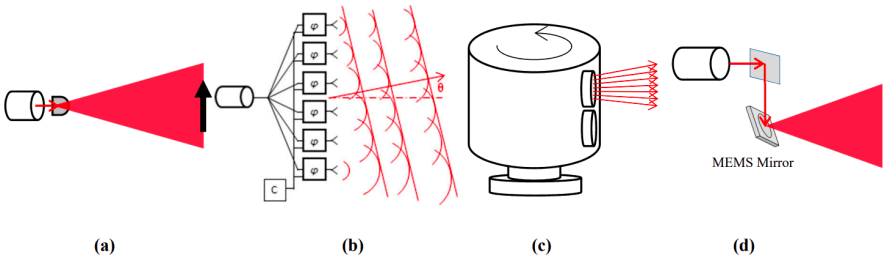


Figure 1-4: (a) Flash LiDAR (b) Optical Phase Array scanning LiDAR (c) Mechanical scanning LiDAR (d) MEMS based scanning LiDAR [19].

Non-scanning LiDAR or "Flash" LiDAR entirely illuminates its FOV with the laser at once, while scanning LiDAR covers the FOV piece by piece.

Flash LiDAR suffers from a low signal-to-noise ratio (SNR) because only a small fraction of the send out light is returned to its receivers because the laser is not focused during emission [19]. This limits its measurement range or requires a very high power laser [19].
Scanning LiDAR can achieve a higher signal to noise ratio because of the focus of the lidar pulses and is therefore often preferred when doing SLAM [19].
Scanning LiDAR systems steer the laser beam over time to cover the FOV. The most popular kind of scanning LiDARs do this mechanically through use of a motor. A good example of a motorized mechanical LiDAR scanner is the Velodyne VLP-16.
The Velodyne consists of an array of vertically stacked emitters and receivers that are rotated along an axis by a motor (Fig. 1-4 c).
Recently a new branch of scanning LiDARs has been introduced that, instead of using a motor, uses a moving mirror based on MEMS technology to let the laser beam scan the FOV (Fig. 1-4 d). The Intel L515 falls in this category.

1-2-2 Differences between the Intel L515 and conventional LiDAR

The Intel L515 utilizes a MEMS mirror for scanning instead of a motor that spins the laser beam. The low power of the laser utilized in the Intel L515 compared to a motorized mechanical LiDAR also has less sensing range. This makes the overall FOV different from a motorized mechanical LiDAR (Fig. 1-5). The Intel has got a maximum range of 9m assuming optimal conditions and covers a viewing angle of 70 degrees. The Velodyne VLP-16 which is a the most commonly researched mechanical LiDAR has got 90m range and 360 degrees FOV clearly outmatching the FOV of the Intel L515.

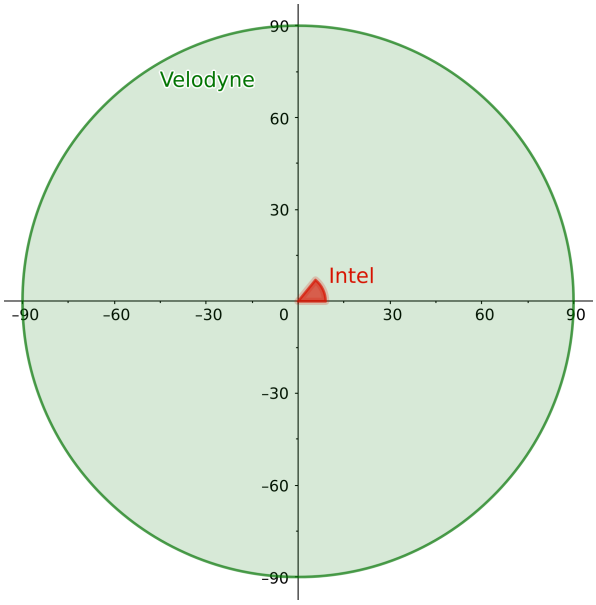
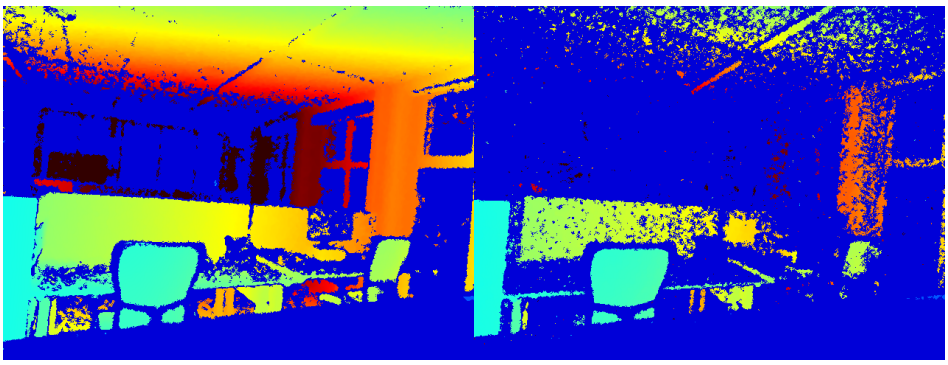


Figure 1-5: Field of view projected onto the x,y plane of the Intel L515 compared to the Velodyne VLP-16. All distances in this figure are in meters.

The small FOV of the Intel L515 can cause the degenerate situtations in which a LiDAR SLAM algorithm is not able to localize itself. Degenerate situations are mostly caused by too

few features, or when all features lie on a plane, causing directions parallel to the plane to be unobservable.

Another difference between the Intel L515 and more expensive LiDARs is its sensitivity to ambient light. If ambient sunlight enters a room through windows, it can cause blackouts in the LiDAR stream (Fig. 1-6).
The blackouts reduce the effective range of the Intel L515 dramatically. This is another problem that needs to be solved to use the Intel L515 for indoor localization. In the scene of Figure 1-6, the lidar depth stream contained only 19 % pixels with depth information when sunlight was entering through windows. At night without the ambient light 54 % of pixels had depth information.



(a) lidar depth stream (night) **(b)** lidar depth stream (day)

Figure 1-6: The input of the L515 lidar when looking at an area with windows on the right side. Pixels for which there is no depth information are dark blue.

The main advantages of using a MEMS based LiDAR like the Intel L515 woud be: lower weight, lower cost and power consumption. This makes the sensor interesting for operation in autonomous robotics or scenario's where the least amount of inference on work is required. The full specs of the Intel L515 compared to the Velodyne VLP-16 are found in Table 1-1.

Table 1-1: Specifications of the Intel L515, a low power MEMS based LiDAR, compared with the Velodyne VLP 16, a widely used motorized mechanical LiDAR.

	Intel L515	**Velodyne VLP16**
Cost	~$ 350	> $ 4000
Field of View	70°x 55 °	360°x 32 °
Range	0.25 - 9 m	0.5-100 m
Accuracy	1.4 cm	3 cm
Update rate	30 Hz	5-20 Hz
Horizontal Resolution	0.07 °	0.1-0.4°
Vertical Resolution	0.07 °	2 °
Dimensions	61 x 26 mm	103 x 72 mm
Weight	95 g	860 g
Power Consumption	< 3.5 W	8 W
Maximum Illuminance	< 500 lux (office light) [10]	> 20.000 lux approx. Anything but direct sunlight [12]

1-2-3 Inertial Measurement Unit

The IMU of the Intel L515 consists of an accelerometer and a gyroscope. The gyroscope measures angular velocity and the accelerometer measures linear acceleration. The sensors can be modelled as a perfect measurement with some added bias and some Gaussian noise (1-2, 1-3)[5].

$$ {}^{B}\boldsymbol{\omega}_m = {}^{B}\boldsymbol{\omega}_r + \mathbf{b}_g + {}^{B}\boldsymbol{\lambda}_\omega \tag{1-2} $$

$$ {}^{B}\boldsymbol{a}_m = {}^{B}\boldsymbol{a}_r + \mathbf{b}_a + {}^{W}\mathbf{g} + {}^{B}\boldsymbol{\lambda}_a \tag{1-3} $$

${}^{B}\boldsymbol{\omega}_m$,${}^{B}\boldsymbol{a}_m$ are the measured angular velocity and linear acceleration. ${}^{B}\boldsymbol{\omega}_r$,${}^{B}\boldsymbol{a}_r$ are the real angular velocity and linear acceleration. ${}^{W}\mathbf{g}$ is the gravity vector and ${}^{B}\boldsymbol{\lambda}_a$,${}^{B}\boldsymbol{\lambda}_\omega$ are additional white noise.

MEMS technology allows to produce small cheap IMUS. The IMU measurements can be modeled as a perfect measurement with additional bias and white noise. Depending on the quality of the sensor these effects can be reduced. To obtain the pose out of IMU measurements the linear acceleration measurements of the accelerometer need to be integrated twice, while the angular velocity measurements need to be integrated once. This process is called dead reckoning. The integration causes drift in the final pose estimate because the white noise present in the signal will accumulate upon integration and impact the final pose estimate. To illustrate the accumulation of drift in dead reckoning, figure 1-7 shows an example of integration of white noise accumulates [13]. As can be seen from Figure 1-7, as the noise gets integrated over 100 samples, it's not unrealistic for the drift (the integrated signal) to be 15 times larger than the original noise variance.

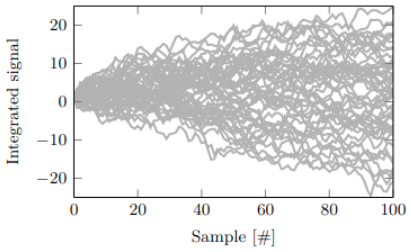


Figure 1-7: Integration of a white noise signal $yt \sim \mathcal{N}(0, 1)$ for 50 noise realizations [13].

1-3 Problem Formulation

LiDAR SLAM is a powerful tool for mapping an unknown environment. In recent years tools have been developed for LiDARs to do real time SLAM [22] [17]. LiDAR SLAM allows for localization and mapping without GNSS, while building very accurate maps due to the accuracy of the LiDAR.
The prohibitive costs of LiDARs and the size of the equipment is however a drawback keeping the technology from widespread use for situational awareness in emergencies.
The Intel L515 is an interesting device to use for LiDAR SLAM because it bypasses these drawbacks with its small size and weight, low power consumption and low cost.
The Intel L515 does have some problems that prevent it from being used for situational awareness, namely a large sensitivity to ambient (sun) light and a higher chance to end up in a degenerate scenario because of its small FOV.
The IMU of the Intel L515 should be able to counter these effects to some extend, since it doesn't depend on the availability of nearby features nor ambient light.
To fuse the IMU measurements with the LiDAR odometry estimations, a graph based approach is chosen that is customized for use with the Intel L515.
In this thesis the problems plaguing the LiDAR are researched and graph based sensor fusion with an IMU is proposed as a solution.

The following main research question is chosen to verify the research:

What is the added benefit of adding an IMU to localization of the Intel L515 when performing LiDAR SLAM?

This question is divided into three sub questions:´

1. How will the IMU affect localization drift?

2. What is the benefit of the IMU when ambient light is present?

3. To what extent can the IMU be used to reduce the effect of degenerate LiDAR scenario's?

1-4 Outline of the Thesis

This chapter covered the most relevant topics and challenges for using the Intel L515 for LiDAR SLAM. Chapter 2 of this thesis focuses on the LiDAR localization pipeline in detail. Chapter 3 describins the main findings of this thesis on the applicabilaty of the Intel L515 for doing LiDAR inertial SLAM in the form of a manuscript. Chapter 4 concludes the findings and provides an outlook on follow up research on this topic.

Chapter 2

Background Theory

This chapter introduces the background theory that is necessary for doing graph based lidar inertial localization. First the motion model of the sensor is derived when the sensor is moving through the world, along with the state description of the sensor (Sec. 2-1).
Then the factor graph model of our state estimation problem is introduced Sec 2-2. The factor graph model is a visualization of the overall optimization problem of finding the locations of the sensor at different points in time.
To find the most likely set of states of the sensor, the factor graph problem needs to be defined in terms of input measurements from the Light Detection and Ranging (LiDAR) and Inertial Measurement Unit (IMU). The problem formulation for the IMU is described in Sec. 2-3, followed by the problem formulation for the LiDAR in Sec. 2-4.
The final step for finding the locations of the sensor at different points in time and therefore solving the localization problem comes after the cost function for the graph based optimization is created with the measurement models. When the cost function is created, it needs to be minimized to obtain the optimal set of states that match the measurements. This needs to be done in a computationally efficient way, which is described in Sec. 2-5.

2-1 Motion Model

An object moving through three dimensional space has got six degrees of freedom. The position of the object has three degrees of freedom. The orientation of the object has also got three degrees of freedom. The combination of a position and orientation is called a pose. The pose of an object can be described by fixing a coordinate frame to the object and relating it to another coordinate frame.
In the case of a sensor moving through an unknown environment, the pose can by described by fixing the body coordinate frame B to a point on the sensor and then relating it to the world coordinate frame W that is fixed in the environment.
At least six coordinates are needed to describe this relation: three for the position and three for the orientation.

One of the ways to describe the pose of B in W is by using the Euclidean distance between the centers of B and W to describe the position, combined with Euler angles to define the rotation. In that case, the position of the sensor is described by the Cartesian coordinates $x, y, z \in \mathbb{R}^3$ while the orientation is described by the angles $\alpha, \beta, \gamma \in SO(3)$. Here $SO(3)$ is the 3D rotation group.
Mapping any point from B to W can be done by doing a translation t and applying a rotation R (2-1).

$$ {}^{W}\mathbf{p} = \underbrace{\begin{bmatrix} \cos\gamma\cos\beta & -\sin\gamma\cos\alpha + \cos\gamma\sin\beta\sin\alpha & \sin\gamma\sin\alpha + \cos\gamma\sin\beta\cos\alpha \\ \sin\gamma\cos\beta & \cos\gamma\cos\alpha + \sin\gamma\sin\beta\sin\alpha & -\cos\gamma\sin\alpha + \sin\gamma\sin\beta\cos\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix}}_{{}^{W}R_B} {}^{B}\mathbf{p} + {}^{W}\mathbf{t}_B \tag{2-1} $$

Here ${}^{W}\mathbf{p}$ is a vector containing the coordinates of a point in the world, ${}^{W}R_B$ is the rotation matrix that maps the body frame to the world frame. ${}^{W}\mathbf{t}_B$ is the translation vector containing the x, y, z coordinates of the center of B expressed in the W-frame. α, β, γ are the rotations of the axis following Euler rotations. The R matrix can also be defined differently by using other representations like for example quaternions, or Tait-Bryan angles. As long as the R matrix that follows from the representation doesn't violate the rules of the $SO(3)$ group, it's a valid representation of the orientation.

2-1-1 Homogeneous Transformation

To speed up the transformation of a point between coordinates frames, one can use homogenous coordinates. The homogeneous coordinates of a point (X_c, Y_c, Z_c) in Cartesian coordinates can be obtained by adding a 1 as a fourth coordinate (i.e. $(X_c, Y_c, Z_c, 1)$).
By using homogeneous coordinates, the transformation of a point from one coordinate frame to another can be done with a single multiplication (2-2) instead of a multiplication and addition as is the case in Eq. 2-1.

$$ \begin{bmatrix} {}^{W}\mathbf{p} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} {}^{W}R_B & {}^{W}\mathbf{t}_B \\ \mathbf{0} & 1 \end{bmatrix}}_{{}^{W}H_B} \begin{bmatrix} {}^{B}\mathbf{p} \\ 1 \end{bmatrix} \tag{2-2} $$

Here ${}^{W}H_B$ is the homogenous transformation matrix.

2-1-2 State of the Sensor

The localization problem of the sensor can be phrased as a state estimation problem. In the case of the Intel® RealSense™ LiDAR Camera L515 (Intel L515) sensor, the state at time i is a combination of the orientation ${}^{W}R_B^T(i)$, position ${}^{W}\mathbf{p}^T(i)$, linear velocity ${}^{W}\mathbf{v}^T(i)$, the accelerometer bias $({}^{B}\mathbf{b}_a^T(i))$ and gyroscope bias ${}^{B}\mathbf{b}_g^T(i)$ (2-3).

$$ \mathbf{x}(i) = [{}^{W}R_B^T(i), {}^{W}\,\mathbf{p}^T(i), {}^{W}\,\mathbf{v}^T(i), {}^{B}\,\mathbf{b}_a^T(i), {}^{B}\,\mathbf{b}_g^T(i)] \tag{2-3} $$

2-2 Graph Based Optimization

To obtain the states of the sensor at different points in time a graph based optimization needs to be done. The optimization problem can be visualized as a factor graph (Fig. 2-1). The factor graph provides an intuitive way of understanding the optimization problem at hand.

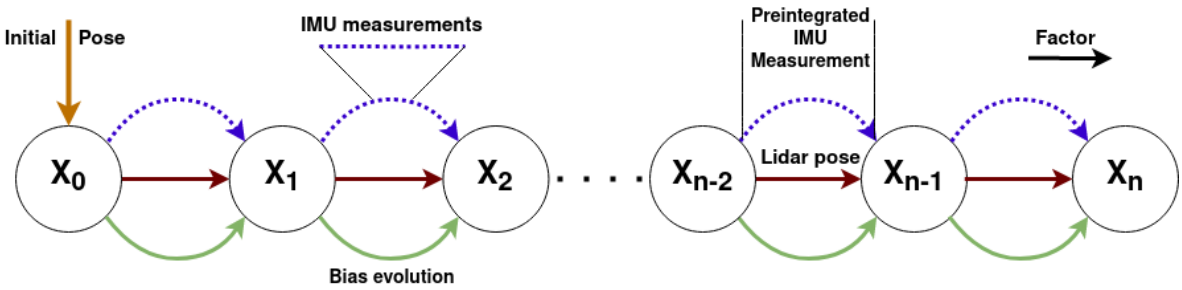
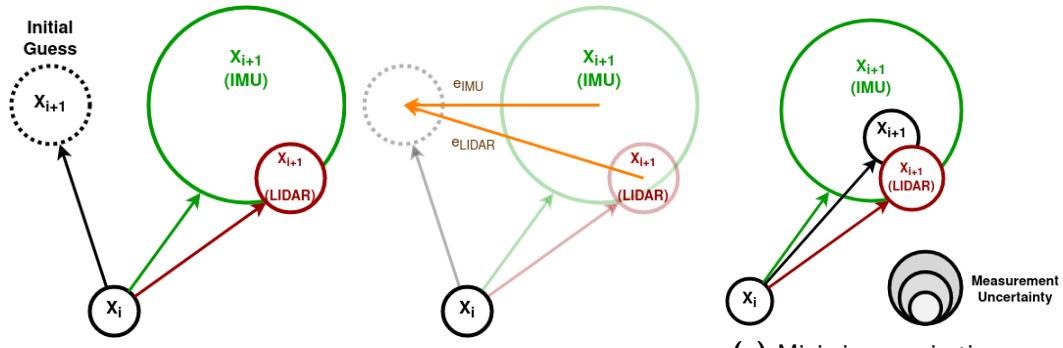


Figure 2-1: Visualization of the factor graph of the graph based optimization performed in this thesis.

The variables of the optimization are the nodes of the graph. These nodes are visualized by the circles in Fig. 2-1.
The nodes are constrained by their relations to other nodes in the graph. These relations are the factors of the graph and are visualized by arrows in Fig. 2-1.
In the factor graph model of the optimization done in this thesis only sequential factors are present i.e. factors that relate sequential nodes. It is also possible to create loop closures in the model by creating non-sequential factors, but this is deemed out of scope for answering the research questions.
The sequential factors in our model consist of preintegrated IMU measurements, LiDAR pose measurements and bias evolution changes.
For each of the factors a reprojection error is defined (Fig. 2-2). The reprojection error is defined as the difference between the current estimate of a node of the graph and the estimated location by the factors.



(a) Initialize estimate **(b)** Define reprojection error **(c)** Minimize reprojection error

Figure 2-2: The three step process to solve a factor graph with 2 nodes.

After the reprojection error is defined for all the factors in the graph, they can be combined to create a cost function. This cost function can be solved by a graph based optimization

algorithm to find the optimal configuration of the nodes. The cost function used in this thesis consists of the reprojection errors of all the factors (2-4) [5].

$$\mathbf{X}^*_{1:n} \rightarrow \underset{\mathbf{x}}{argmin} \ \ \|\mathbf{r}_0(x)\|^2_{\Sigma_0} + \sum_{x=0}^{n} \left(\|\mathbf{r}_L(x)\|^2_{\Sigma_L} + \|\mathbf{r}_I(x)\|^2_{\Sigma_I} + \|\mathbf{r}_b(x)\|^2_{\Sigma_b} \right) \tag{2-4}$$

Here $\mathbf{r}_0(x)$ is the initial pose estimate error, $\mathbf{r}_L(x)$ is the LiDAR estimate reprojection error, $\mathbf{r}_I(x)$ is the IMU preintegration reprojection error, and $\mathbf{r}_b(x)$ is the bias reprojection error. Each of these reprojection errors is weighted by their respective covariance Σ as formulated in Formula 2-5.

$$\|\mathbf{r}_x(i)\|^2_{\Sigma_x} = \mathbf{r}_x(i)^T \Sigma_x \mathbf{r}_i(x) \tag{2-5}$$

The IMU preintegration reprojection error is explained in Sec. 2-3. The lidar movement estimation reprojection error is explained in Sec. 2-4. The bias evolution reprojection error is explained in Sec. 2-3-3.
The graph based optimization algorithm used for the sensor fusion in this thesis is then explain in Sec. 2-5.

2-3 IMU preintegration

The IMU factors are constructed by using IMU preintegration [5]. IMU preintegration allows combining a large amount of IMU measurements into a single measurement: a preintegrated IMU measurement.
This is desirable because IMU measurements have a much faster update frequency than lidar measurements. By updating the factor graph only at new lidar estimates and using Preintegrated IMU measurements to summarize all the measurements between two lidar estimates, the graph becomes smaller than when updating the graph for every IMU measurement. The smaller graph allows for real time localization.
Preintegrated IMU measurements also have an advantage over normally integrating the IMU measurements between two lidar estimates. A Preintegrated IMU measurement between two lidar estimates is namely described in such a way that it avoids having to iterate through every single IMU measurement again when the values of the nodes change. In the case of normally integrating the IMU measurements this is not the case.

2-3-1 Summarizing the IMU measurments

In this section the process of creating an IMU preintegration factor is derived, starting with the IMU measurement model (2-6, 2-7) [5].

$${}^B\boldsymbol{\omega}_m = {}^B\boldsymbol{\omega}_r + \mathbf{b}_g + {}^B\boldsymbol{\lambda}_\omega \tag{2-6}$$

$${}^B\boldsymbol{a}_m = {}^B\boldsymbol{a}_r + \mathbf{b}_a + {}^W\mathbf{g} + {}^B\boldsymbol{\lambda}_a \tag{2-7}$$

Here $\boldsymbol{\omega}_r$ and $\boldsymbol{a}_r$ are the real angular velocity and linear acceleration. They consist of respectively the measurements $\boldsymbol{\omega}_m$ and $\boldsymbol{a}_m$, the biases of the sensor $\mathbf{b}_a, \mathbf{b}_g$ and additional gaussian noise $\boldsymbol{\lambda_\omega}, \boldsymbol{\lambda_a}$.
In this section it is assumed that the bias of the sensors is preset at a certain value, and does not change over time. In Sec. 2-3-2 the model is updated to incorporate bias changes. The preset bias for the accelerometer and gyroscope at time i are written with $\bar{\mathbf{b}}_a(i)$ and $\bar{\mathbf{b}}_g(i)$ respectively. $\boldsymbol{\omega}_r$ and $\boldsymbol{a}_r$ can be integrated (twice in the case of $\boldsymbol{a}_r$) to obtain the orientation and position of the sensor at the next measurement (2-8-2-10)[5].

$$^W R_B(i+1) = ^W R_B(i)\text{Exp}(^B\boldsymbol{\omega}_r(i)\Delta t) \tag{2-8}$$
$$^W \mathbf{v}(i+1) = ^W \mathbf{v}(i) + ^W \mathbf{a}_r(i)\Delta t \tag{2-9}$$
$$^W \mathbf{p}(i+1) = ^W \mathbf{p}(i) + ^W \mathbf{v}(i)\Delta t + \frac{1}{2}\, ^W \mathbf{a}_r(i)\Delta t^2 \tag{2-10}$$

Here Δt is the time difference between the two IMU measurements.
By substituting (2-6, 2-7) into (2-8 - 2-10), and transforming $^W\mathbf{a}_r(i)$ to the B frame, the equation relating the measurements to state updates is obtained (2-11-2-13).

$$^W R_B(i+1) = ^W R_B(i)\text{Exp}((\boldsymbol{\omega}_m(i) - \bar{\mathbf{b}}_g(i) - \boldsymbol{\lambda}_\omega(i))\Delta t) \tag{2-11}$$
$$\begin{aligned}\mathbf{v}(i+1) = \mathbf{v}(i) &+ \mathbf{g}\Delta t \\ &+ R(i)(^B\mathbf{a}_m(i) - \bar{\mathbf{b}}_a(i) - \boldsymbol{\lambda}_a(i))\Delta t\end{aligned} \tag{2-12}$$
$$\begin{aligned}\mathbf{p}(i+1) = \mathbf{p}(i) + \mathbf{v}(i) &+ \frac{1}{2}\mathbf{g}\ \Delta t^2 \\ &+ \frac{1}{2}\ ^W R_B(i)(^B\mathbf{a}_m(i) - \bar{\mathbf{b}}_a(i) - \boldsymbol{\lambda}_a(i))\Delta t^2\end{aligned} \tag{2-13}$$

Now that the update equation between two consecutive IMU measurements is created, a formulation needs to be derived that describes the state update between two arbitrary time points.
The relative movements $\Delta\mathbf{v}_{ij}, \Delta\mathbf{p}_{ij}, \Delta R_{ij}$ (2-14 - 2-16) can be found by iterating (2-11-2-13), and compensating for the initial state.

$$\begin{aligned}\Delta\mathbf{p}_{ij} &= R_i^T\Big(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2\Big) \\ &= \sum_{k=i}^{j-1}\Big(\frac{3}{2}\Delta R_{ik}\big(\mathbf{a}_m(k) - \bar{\mathbf{b}}_a(k) - \boldsymbol{\lambda}_a(k)\big)\Delta t^2\Big)\end{aligned} \tag{2-14}$$
$$\begin{aligned}\Delta\mathbf{v}_{ij} &= R_i^T(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) \\ &= \sum_{k=i}^{j-1}\Delta R_{ik}\big(\mathbf{a}_m(k) - \bar{\mathbf{b}}_a(k) - \boldsymbol{\lambda}_a(k)\big)\Delta t\end{aligned} \tag{2-15}$$
$$\begin{aligned}\Delta R_{ij} &= R_i^T R_j \\ &= \prod_{k=i}^{j-1}\text{Exp}\Big((\omega_m(k) - \bar{\mathbf{b}}_g(k) - \boldsymbol{\lambda}_\omega(k))\Delta t\Big)\end{aligned} \tag{2-16}$$

With $\Delta\mathbf{v}_{ij}, \Delta\mathbf{p}_{ij}, \Delta R_{ij}$ the difference of the state values between the state at timestamp i and any arbitrary future state j.

The equations on the top lines of (2-14-2-16) are dependant on the nodes of the factor graph. The equation on the bottom lines of (2-14-2-16) only depend on the measurements, and are therefore independent of the states in the graph. This is critical, because this means that the bottom parts only have to be computed once, when doing graph based optimization.
The residuals errors are defined as the difference between the two lines of each equation (2-17 - 2-19).

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{p}_{ij}} &= R_i^T\Big(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2\Big) \\ &\quad - \Delta\mathbf{p}_{ij} - \delta\mathbf{p}_{ij}(b)\end{aligned} \tag{2-17}$$

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{v}_{ij}} &= R_i^T(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) \\ &\quad - \Delta\mathbf{v}_{ij} + \delta\mathbf{v}_{ij}(b)\end{aligned} \tag{2-18}$$

$$\mathbf{r}_{\Delta R_{ij}} = \mathrm{Log}\Big((\Delta R_{ij}\mathrm{Exp}(\delta R_{ij}(b)))^T R_i^T R_j\Big) \tag{2-19}$$

$$\mathbf{r}_I(x) = \begin{bmatrix}\mathbf{r}_{\Delta\mathbf{p}_{ij}} & \mathbf{r}_{\Delta R_{ij}} & \mathbf{r}_{\Delta\mathbf{v}_{ij}} & \mathbf{0}\end{bmatrix}^T \tag{2-20}$$

Where $\mathbf{r}_{\Delta\mathbf{p}_{ij}}, \mathbf{r}_{\Delta\mathbf{v}_{ij}}, \mathbf{r}_{\Delta R_{ij}}$ is the position, velocity and orientation reprojection error for IMU preintegration. The IMU preintegration reprojection error (2-20) can now be formed, which is inserted in the graph based cost function for all sensors (2-4).

2-3-2 Bias model updating

In section 2-3-1 the bias was assumed to be constant. Because the bias is also part of the state (2-3), the model needs to be extended to incorporate bias changes. In [5] a model for adding a perturbation to the bias is derived. This model leads to the reprojection errors that are needed in the factor graph model (2-21, 2-22, 2-23)[5].

$$\begin{aligned}r_{\Delta\mathbf{p}_{ij}} &= R_i^T\Big(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2\Big) \\ &\quad - [\Delta\mathbf{p}_{ij}(\bar{\mathbf{b}}^a(i), \bar{\mathbf{b}}^g(i)) + \frac{\partial\Delta\bar{\mathbf{p}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g + \frac{\partial\Delta\bar{\mathbf{p}}_{ij}}{\partial\mathbf{b}^a}\delta\mathbf{b}^a]\end{aligned} \tag{2-21}$$

$$\begin{aligned}r_{\Delta\mathbf{v}_{ij}} &= R_i^T(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) \\ &\quad - [\Delta\mathbf{v}_{ij}(\bar{\mathbf{b}}^a(i), \bar{\mathbf{b}}^g(i)) + \frac{\partial\Delta\bar{\mathbf{v}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g + \frac{\partial\Delta\bar{\mathbf{v}}_{ij}}{\partial\mathbf{b}^a}\delta\mathbf{b}^a]\end{aligned} \tag{2-22}$$

$$r_{\Delta R_{ij}} = \mathrm{Log}\Big((\Delta R_{ij}(\bar{\mathbf{b}}^g(i))\ \mathrm{Exp}(\frac{\partial\Delta\bar{R}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g))^T R_i^T R_j\Big) \tag{2-23}$$

Here the Jacobians $\{\ \frac{\partial\Delta\bar{\mathbf{p}}_{ij}}{\partial\mathbf{b}^g}, \frac{\partial\Delta\bar{\mathbf{v}}_{ij}}{\partial\mathbf{b}^g}, \frac{\partial\Delta\bar{R}_{ij}}{\partial\mathbf{b}^g}\ \}$ are calculated when the factor is created, avoiding recalculation as the bias changes. The $\delta\mathbf{b}^g$ and $\delta\mathbf{b}^a$ are defined as the differences between the bias according to the state estimation and the bias at the linearization point (2-24, 2-25).

$$\delta\mathbf{b}^g = \mathbf{b}^g - \bar{\mathbf{b}}^g \tag{2-24}$$

$$\delta\mathbf{b}^a = \mathbf{b}^a - \bar{\mathbf{b}}^a \tag{2-25}$$

2-3-3 Bias Factor

The bias is assumed to change like a Brownian motion between two state updates (2-26 - 2-27) [5].

$$\mathbf{b}^a(i+1) = \mathbf{b}^a(i) + \eta_a(i) \tag{2-26}$$
$$\mathbf{b}^g(i+1) = \mathbf{b}^g(i) + \eta_g(i) \tag{2-27}$$

With $\eta_g = \Delta t_{i,i+1}\text{Cov}(\eta_{gd})$ and $\eta_a = \Delta t_{i,i+1}\text{Cov}(\eta_{ad})$. η_{ad}, η_{gd} are properties of the IMU, and are defined as a zero mean random process with covariance equal to η_{gd}, η_{ad}.
The bias change is zero mean Gaussian. The reprojection error for this model is simply the difference between two subsequent bias values, weighted by their respective covariance matrices (2-28)[5].

$$\|\mathbf{r}_b(i)\|^2_{\Sigma b} = \|\mathbf{b}^g(j) - \mathbf{b}^g(i)\|^2_{\eta_g} + \|\mathbf{b}^a(j) - \mathbf{b}^a(i)\|^2_{\eta_g} \tag{2-28}$$

This bias reprojection error $\mathbf{r}_b(i)$ is then added to the cost function for all sensors (2-4).

2-4 Lidar Odometry and Mapping

The method used in this thesis for obtaining the lidar movement is Solid State Lidar Simultaneous Localization and Mapping (SSL-SLAM) [20]. SSL-SLAM is based on LiDAR Odometry And Mapping (LOAM) [22], but modified for use with the Intel L515 instead of a mechanical LiDAR. SSL-SLAM, like LOAM is a feature based LiDAR Simultaneous Localization And Mapping (SLAM) algorithm. Feature based LiDAR SLAM algorithms provide better performance and computational speed than point based methods that match points directly [9]. This allows for real time localization.
SSL-SLAM, like LOAM, uses a two step process to estimate motion out of the LiDAR point cloud scan.
First edge and plane feature points are extracted from the unordered pointcloud (Sec. 2-4-1). Then, the relative transformation of the sensor is found iteratively by matching these points to previously observed planes and edges and minimizing a cost function based on the distance of the observed feature points to their matched counterparts (Sec. 2-4-2).
This relative motion estimate is then used to define the lidar factor in the factor graph (Sec. 2-4-3).

2-4-1 Lidar Edge/Plane Detection

Edge and plane point detection is the first step for obtaining usable features out of the unordered pointclouds. Because the Intel L515 scans a plane instead of a line like a mechanical lidar, the edge/plane detection metric of LOAM [22] cannot be used. The edge/plane metric of SSL-SLAM [20] is based on scanning a plane, and is therefore used.
In this section the edge/plane detection of SSL-SLAM [20] is explained.
Let $\mathcal{P}_k$ be the set of LiDAR points that is obtained from the LiDAR at time k. To obtain the

feature points, $\mathcal{P}_k$ is first segmented into smaller blocks. Let $\mathbf{p}_k^{i,j}$ be a point in $\mathcal{P}_k$ with x, y, z coordinates expressed in the body frame of the sensor. The vertical angle $\alpha_{i,j}$ and horizontal angle $\theta_{i,j}$ between the point and the optical axis of the lidar is then defined (2-29, 2-30) [20].

$$\alpha_{i,j} = \arctan\Big(\frac{y_{i,j}}{x_{i,j}}\Big) \tag{2-29}$$

$$\theta_{i,j} = \arctan\Big(\frac{z_{i,j}}{x_{i,j}}\Big) \tag{2-30}$$

The points in $\mathcal{P}_k$ are segmented based on their $\alpha_{i,j}$ and $\theta_{i,j}$ into $M \times N$ blocks. The segmentation is done by splitting the interval between the lowest vertical angle α_{min} and the highest vertical angle α_{max} into M parts that each cover an angle range of $\alpha_{res} = \frac{\alpha_{max} - \alpha_{min}}{M}$. Let m be the vertical cell number of one of the blocks, then all the points for which $m = 1$ lie in interval $[\alpha_{min}, \alpha_{min} + \alpha_{res}]$, and the points for which $m = 4$ for example lie in interval $[\alpha_{min} + 3\alpha_{res}, \alpha_{min} + 4\alpha_{res}]$.
The horizontal interval is similarly split into N parts between the lowest horizontal angle θ_{min} and the highest horizontal angle θ_{max}. Each of these N parts cover an angle range of $\frac{\theta_{max} - \theta_{min}}{N}$.
Let n be the horizontal cell number of each block, then each point $\mathbf{p}_k^{(i,j)}$ can be assigned to a block m, n based on $\alpha_{i,j}, \theta_{i,j}$.
After all points $\mathbf{p}_k^{i,j} \in \mathcal{P}_k$ are added to their respective (m, n), the geometric center of each cell $\mathbf{p}_k^{(m,n)}$. This geometric center is then used to create the smoothness metric $\sigma_k^{m,n}$ (2-31)[20] for each (m, n).

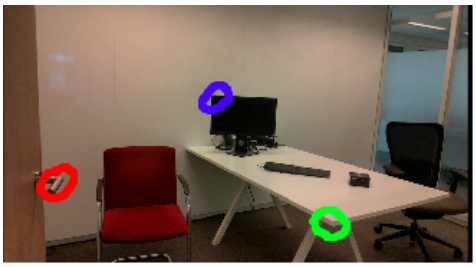
$$\sigma_k^{m,n} = \frac{1}{\lambda^2} \cdot \sum_{\mathbf{p}_k^{(i,j)} \in \mathcal{S}_k^{(m,n)}} (||\mathbf{p}_k^{(i,j)}||_2^2 - ||\mathbf{p}_k^{(m,n)}||_2^2) \tag{2-31}$$

with

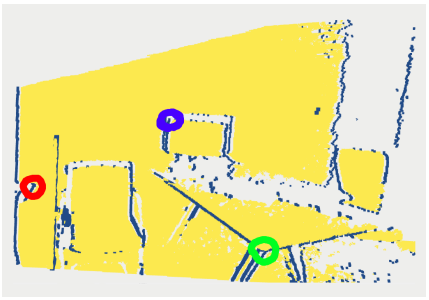
$$\mathcal{S}_k^{(m,n)} = \{\ \mathbf{p}_k^{i,j} \mid i \in [m - \lambda, m + \lambda],\ j \in [n - \lambda, n + \lambda]\} \tag{2-32}$$

Here λ is a pre defined searching radius.

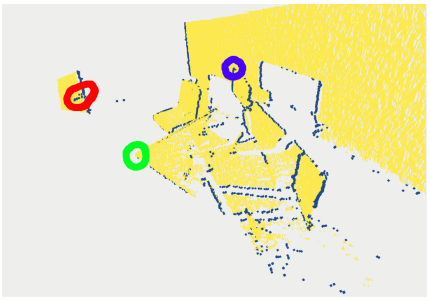
In the case that the points in a cell (m, n) lie on a plane, the values of the points $\mathbf{p}_k^{(i,j)} \in \mathcal{S}_k^{(m,n)}$ are distributed approximately equal around $\mathbf{p}_k^{(m,n)}$. A low value for the sum in (2-31) is then achieved because the points farther away from $\mathbf{p}_k^{(m,n)}$ in (2-31) counterbalance each other due to the symmetry of the plane. This causes a low $\sigma_k^{m,n}$.
If $\sigma_k^{m,n}$ lies below a threshold, then the corresponding $\mathbf{p}_k^{(m,n)}$ is marked as a plane point.
If an edge is present in a cell (m, n), a high $\sigma_k^{m,n}$ is achieved. In that case the value for the sum of (2-31) is maximized because of the edge geometry.
If $\sigma_k^{m,n}$ is larger than a threshold, the corresponding $\mathbf{p}_k^{(m,n)}$ is marked as an edge point.
The result of filtering out the plane and edge points of an unordered point cloud is shown in Fig. 2-3.



(a) Camera Input



(b) Filtered pointcloud (camera viewpoint)



(c) Filtered pointcloud (another viewpoint)

Figure 2-3: Example of filtering a point cloud obtained with the Intel L515 with edge/plane detection. (a) The visual input of the scene. (b,c) The filtered edge (blue) and plane (yellow) points of the scene from two different viewing angles.

2-4-2 Lidar Movement Estimation

SSL-SLAM finds the relative lidar movement by creating a cost function based on the distance between the observed feature points and previously observed features. The distance between the observed feature points and previously observed features is then minimized to find the optimal pose.
The first step of this process is transforming the observed feature points to the same coordinate frame as the previously observed features. The previously observed features are stored in the world coordinate frame. The features in a new scan k are projected to this coordinate frame as a function of the lidar pose (2-33).

$$\mathbf{p}_{k,W}^{i,j} = \mathbf{T}_k \ \mathbf{p}_k^{i,j} \tag{2-33}$$

With $\mathbf{T}_k$ the homogenous transformation matrix that translates $\mathbf{p}_k^{i,j}$ from the lidar frame to the world frame based on lidar pose.

The feature point locations are initialized by projecting the previous lidar movement into the future (2-34) to obtain an initial estimate for $\mathbf{p}_{k,W}^{i,j}$.

$$\mathbf{T}_{k,init} = \mathbf{T}_{k-1}(\mathbf{T}_{k-2}^{-1}\mathbf{T}_{k-1}) \tag{2-34}$$

Here $\mathbf{T}_{k-1}$ and $\mathbf{T}_{k-2}$ are the homogeneous transformation matrices of the previous two poses. With this initial pose estimate the feature points are projected to the world frame using (2-33).

For each of the projected feature points in the world frame, the closest corresponding feature points from previous scans are found.
In the case of edge points, the two closest edge points $\mathbf{p}_1, \mathbf{p}_2$ in the local map are found. For each of the feature points, these close corresponding matches are used to conjure a cost function for the optimization.
In the case of edge points the two closest edge points in the local map are used to construct a line. The distance of the newly observed edge point to this line is then defined as a function of the pose of the sensor to build the error metric (2-35).

$$f_{\mathcal{E}}(\hat{\mathbf{p}}_k) = \frac{|(\hat{\mathbf{p}}_k - \mathbf{p}_2) \times (\hat{\mathbf{p}}_k - \mathbf{p}_1)|}{|(\mathbf{p}_2 - \mathbf{p}_1)|} \tag{2-35}$$

With $\hat{\mathbf{p}}_k = \mathbf{T}_k \ \mathbf{p}_k^{i,j}$.

In the case of a plane point, the three closest plane points in the local map are used to define a plane. The distance of the newly observed plane point to this plane is then defined as a function of the pose of the sensor to build the error metric for the plane point (2-36.

$$f_{\mathcal{P}}(\hat{\mathbf{p}}_k) = |(\hat{\mathbf{p}}_k - \mathbf{p}_1)^T \cdot \frac{(\mathbf{p}_1 - \mathbf{p}_2) \times (\mathbf{p}_1 - \mathbf{p}_3)}{|(\mathbf{p}_1 - \mathbf{p}_2) \times (\mathbf{p}_1 - \mathbf{p}_3)|}| \tag{2-36}$$

With $\hat{\mathbf{p}}_k = \mathbf{T}_k \ \mathbf{p}_k^{i,j}$.

The lidar pose $\hat{\mathbf{T}}_k$ is found by stacking (2-35) and (2-36) for every observed edge and plane feature and minimizing the cost (2-37) [20].

$$\hat{\mathbf{T}}_k \rightarrow \arg\min_{\mathbf{T_k}} \sum f_{\mathcal{E}}(\hat{\mathbf{p}}_k) + f_{\mathcal{P}}(\hat{\mathbf{p}}_k) \tag{2-37}$$

This process is done iteratively. At every iteration after the first, the newly found $\hat{\mathbf{T}}_k$ is used to project the feature points at k to the world frame. This can cause changes in the feature point matches in the local map that are closest to the feature points in k, resulting in updates of the cost function.

2-4-3 Defining the Lidar Factor

The relative movement estimations of SSL-SLAM are prone to outliers. The outliers are caused by degenerate scenarios, either due to the small Field of View (FOV) of the sensor or because of ambient light limiting the range of the sensor.
To counter the effects of outliers, the relative movement estimates of the ssl slam algorithm are compared to the motion predictions of the IMU. If the LiDAR predictions lie within a threshold of the Imu prediction, the lidar prediction is classified as an inlier and given a low uncertainty in the cost function (2-4). In the case the LiDAR estimate exceeds the threshold, it is classified as an outlier and given a high uncertainty value in the cost function (2-4).
The LiDAR factor is defined as the difference between the relative lidar pose estimate (2-37) and the pose change between the related nodes. The noise for the lidar factor is assumed to be Gaussian.

2-5 Incremental Smoothing and Mapping

The ISAM2 [11] algorithm is used to solve the factor graph. ISAM2 [11] is used over solvers like Gauss Newton or Levenberg Marquardt due to speed.
The impact of adding a new state to the factor graph as a new LiDAR or IMU preintegration measurement is obtained, is local i.e. only recent states are affected.
ISAM2 introduces a new data data structure called the Bayes tree. The Bayes tree encodes the probability density functions of the states in the factor graph.
ISAM2 leverages the local effect of new measurements, and only updates variables of the state that are affected by a new measurement. Real time performance is achieved by performing only these partial state updates as new measurements are added instead of solving the full optimization problem.

Chapter 3

Manuscript

This chapter presents the main findings of the research as an unpublished manuscript.

Graph Based Lidar Inertial Sensor Fusion for Localization with a Low Range Solid State LiDAR

Arjan Vonk
Delft University Of Technology,
2628 CD Delft, The Netherlands

***Abstract*— Using Mechanical Light Detection And Ranging (LiDAR) to map an environment is currently the most accurate and well deserved go-to method.**
Recently a new branch of LiDARs: Solid State LiDAR's have entered the market. Solid State LiDARs are interesting to use for robotics and real time indoor localization and mapping purposes because of their cheap cost and small size.
In this paper a graph based localization algorithm is proposed to improve the positioning accuracy of a short range solid state LiDAR: The Intel L515. The proposed algorithm fuses location estimates of an Inertial Measurement Unit (IMU) with location estimates by LiDAR scans to improve the localization accuracy. Outlier detection is performed with the IMU to minimize the effect of faulty LiDAR estimates caused by the short range of the LiDAR. The optimization is solved incrementally and runs in real time. The algorithm improves the localization robustness and accuracy compared to the current state of the art for the Intel L515. The addition of the IMU especially reduces altitude drift in Z-direction.

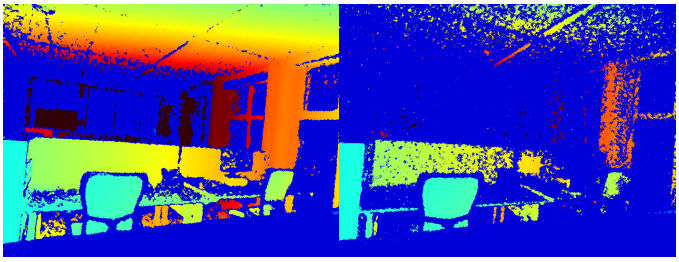
I. INTRODUCTION

Getting a reliable 3-dimensional overview of an emergency situation in real time can be critical for firefighters or police officers on duty [22]. Be it manually or through the use of robots, a 3-dimensional overview of an emergency situation allows better decision making and provides more feedback in situations where a mistake can cost lives [22].
These emergency situations are often indoors, where GNSS is unavailable. To not interfere with the operations it's also required that tools are small and attachable [23].
The process of positioning yourself and mapping an unknown environment with moving sensors is called Simultaneous Localization and Mapping (SLAM).
SLAM has been a widely researched topic since it's inception [30]. The SLAM problem can be summarized as a probabilistic minimization problem (1) [32]:
What is the most likely set of states $\mathbf{x}_{1:t}$ *and environment map* $\mathbf{m}$*, given our inputs* $\mathbf{u}_{1:t}$ *and observations* $\mathbf{z}_{1:t}$*?* .

$$p(\ \mathbf{x}_{1:t},\ \mathbf{m}\ \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{1}$$

In SLAM, the sensors used to obtain the observations of the state can be pre-placed (Bluetooth dongles, security cameras or Wifi) or mobile (Inertial Measurement Units (IMU), cameras, or Light Detection and Ranging (LiDAR) units that are moving through the environment) [23].
The research in this paper focusses on using an IMU and LiDAR to estimate the state in Equation 1 with a recently introduced short range solid state LiDAR: The Intel L515.
The Intel L515 is different from conventional mechanical LiDARs because it uses a Micro Electro Mechanical Systems (MEMS)-mirror to scan the field of view instead of a motor. Using MEMS has got the advantage of faster scanning in a small device, at the cost of decreased laser intensity and Field of View (FoV). The price of the Intel L515 is comparable to the price of a camera, and about 10 times lower than a mechanical LiDAR. This low cost makes the cost comparable to the cost of visual SLAM, it's main competitor, and also attractive for low budget scanning.
Visual SLAM uses a camera to obtain images and then tracks the motion of pixels directly or of observed features in the camera image to retrieve the motion of the camera [15]. Visual SLAM excels at place recognition compared to LiDAR SLAM due to the contrasting textures present in images, that are not present in LiDAR scans.
Visual SLAM algorithms however suffer in low light conditions [19], can suffer from motion blur, and require other sensors or at least two cameras to determine scale. The map created by visual SLAM algorithms is generally sparse and requires triangulation to obtain, because a camera doesn't measure the distance to the features it observes directly.
To create a detailed map, LiDAR is often preferred. The time of flight (ToF) laser sensor of a LiDAR has got higher accuracy, and a larger field of view (FoV) than a camera.
LiDAR based methods are known for being largely invariant to illumination change [10]. This however, is not the case for the Intel L515. Even on a cloudy day, if ambient sunlight comes in through a window, the maximum range of the intel L515 drops to about 2-3 meters instead of the described 9 meter range in the product datasheet (Fig. 1) [28].
Currently there is only one algorithm that performs LiDAR-SLAM with the Intel L515: Solid State LiDAR-SLAM (SSL-SLAM) [2]. SSL-SLAM [2] only uses the LiDAR in its state estimation. The small FoV of the Intel L515, combined with the decreased range from ambient light causes tracking errors with SSL-SLAM [2]. These tracking errors are especially prevalent when doing indoor localization when ambient light is present, or when the scene doesn't contain enough features for localization for a short amount of time.
In this paper the prospects of adding an IMU into the localization process of SSL-SLAM [2] is researched. The goal is to improve the state estimation robustness and posi-

This work was supported by CGI The Netherlands



(a) LiDAR depth stream (night) (b) LiDAR depth stream (day)

Fig. 1: The input of the L515 LiDAR when looking at an area with windows on the right side. The Intel L515 performance drops significantly when ambient (sun)light is present. At night 54 % of pixels contain depth information (a), during the day only 19 % of pixels contain depth information (b). Pixels for which there is no depth information are dark blue.

tioning accuracy. The main contributions of this paper can be summarized as follows:

- A tightly-coupled LiDAR inertial graph based fusion algorithm for the Intel L515 MEMS based solid state LiDAR.
- Incorporating outlier detection for faulty LiDAR measurements on top of SSL-SLAM [2] to improve positioning robustness.
- The algorithm is benchmarked in various ambient light settings, movement conditions and with different feature availability.

II. RELATED WORK

Early LiDAR based localization methods like the Iterative Closest Points (ICP) algorithm focus on directly matching point cloud points to obtain the position [6]. ICP is not suitable for real time LiDAR localization due to rapidly increasing complexity when the number of points increases [33]. It's also prone to end up in local minima, and therefore needs good initialization [33].

To limit the computational complexity, feature based methods have been proposed. One such method is LiDAR Odometry and Mapping (LOAM) [1]. LOAM first extracts edge and plane features from the LiDAR scan. LOAM then minimizes the point-to-line / point-to-plane distance of these points to corresponding features to retrieve the relative motion in real time.

LOAM can also be used with an IMU. In LOAM, the IMU is used only for initialization of the optimization, making the IMU loosely coupled.

Since the conception of LOAM more LiDAR feature based localization and mapping algorithms have been developed that perform the same edge and plane detection steps, together with point-to-line / point-to-plane distance minimization.

LEGO-LOAM [18] for example utilizes LOAM but first filters out the ground plane to estimate height, roll and pitch before estimating the rest of the state.

A current state of the art method that combines LiDAR state estimation with IMU measurements is the Lidar Inertial Odometry via Smoothing and Mapping (LIO-SAM) algorithm [10]. LIO-SAM [10] is a tightly coupled method that utilizes a graph based model to couple the IMU and LiDAR measurements, as well as GPS measurements. Tight coupling of the IMU generally improves localization accuracy [10]. The graph based structure of LIO-SAM also allows for integration of other sensors. LVI-SAM [**?**] is an extension of LIO-SAM that also incorporates a camera into the state estimation that performs visual odometry.

The structure of graph based methods allows for loop closures, which are relations between non sequential states. The graph based structure also makes it easier to incorporate sensors that have different update frequencies with each other.

For MEMS based Solid State LiDARs there exist few algorithms, because of their recent introduction. One Solid State LiDAR algorithm that achieves solid performance is LOAM Livox [7]. LOAM Livox utilizes a solid state Livox LiDAR, that has an angular field of view that is comparable to the Intel L515. The range of the Livox LiDAR however is 260 meters in perfect conditions, which is 25 times higher than the range of the Intel L515.

SSL-SLAM [2] is currently the only LOAM-based algorithm that has been developed on the Intel L515. It has got a modified edge/plane detection metric to compensate for the Intel specifications compared to ordinary LOAM. It is only tested in a warehouse environment without any ambient light, and fails in cases where not enough features are available for scanning because ambient light is present or features are too far from the LiDAR (Section IV).

RTAB map [31] can also be implemented on the Intel L515. RTAB map is a visual method that employs a graph based structure to do loop closures. RTAB map only uses the LiDAR to determine the distance to points observed by the visual algorithm, therefore it's not a full LiDAR SLAM algorithm.

III. METHODOLOGY

An optimization based approach is used to fuse the measurements of the IMU and LiDAR. First the system model and the cost function of the optimization are introduced III-A. Then the individual building blocks of the cost functions are elaborated in III-B - III-D. To conclude the methodology, the optimization is solved III-E.

A. *System Overview*

The state of the sensor consists of the pose of the sensor together with the IMU bias (2).

$$\mathbf{x}(i) = [{}^{W}R_{B}^{T}(i), {}^{W}\mathbf{p}^{T}(i), {}^{W}\mathbf{v}^{T}(i), {}^{B}\mathbf{b}_{a}^{T}(i), {}^{B}\mathbf{b}_{g}^{T}(i)] \quad (2)$$

The IMU frame coincides with the body frame B. B moves relative to the world frame W. ${}^{W}R_{B} \in SO(3)$ represents the rotation matrix that maps B to W. ${}^{W}\mathbf{p}$ represents the position of B in W. ${}^{W}\mathbf{v} \in \mathbb{R}^3$ represents the linear velocity

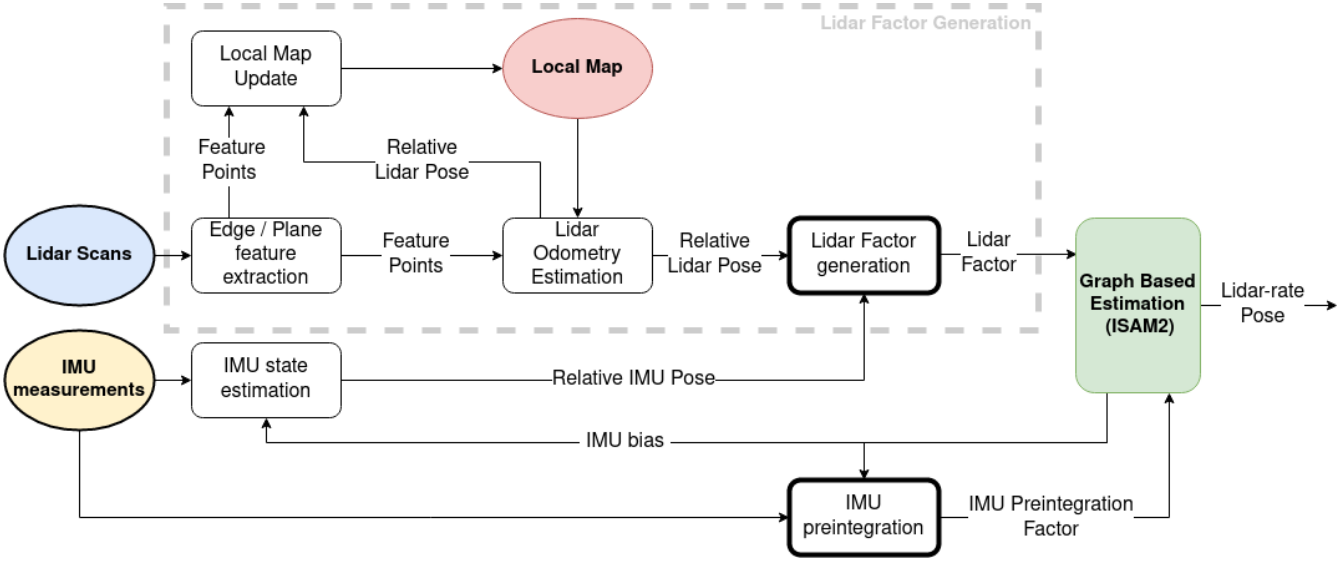


Fig. 2: System model relating the inputs from the IMU and LiDAR to the graph based estimation module

of B in W. ${}^{B}\mathbf{b}_a$ and ${}^{B}\mathbf{b}_g$ represent the bias of the measurements of the accelerometer and gyroscope respectively.
At every new LiDAR pose update i, a state update happens and a new state $\mathbf{x}_i$ is added to the optimization. The evolution of the state is visualized as a factor graph in Figure 3. The cost function for the pose optimization consists of the residual errors of the factors in Figure 3 (3).

$$\mathbf{X}^*_{1:n} \rightarrow \underset{\mathbf{x}}{argmin} \quad \|\mathbf{r}_0(x)\|^2_{\Sigma_0} + \sum_{x=0}^{n} \Big(\|\mathbf{r}_L(x)\|^2_{\Sigma_L} + \|\mathbf{r}_I(x)\|^2_{\Sigma_I} + \|\mathbf{r}_b(x)\|^2_{\Sigma_b} \Big) \tag{3}$$

With:

$$\|\mathbf{r}_x\|^2_{\Sigma_x} = \mathbf{r}_x^T \Sigma_x \mathbf{r}_x \tag{4}$$

Here $\mathbf{r}_0, \mathbf{r}_L, \mathbf{r}_I, \mathbf{r}_b$ are the reprojection errors of the initial pose, LiDAR poses, IMU preintegration measurements, and the bias evaluation. The weights (Σ) given to the errors were determined empirically by a series of experiments (Section IV). The full estimation is done by minimizing the weighted sum (3) [25]. The state is estimated incrementally at every LiDAR update by updating and solving the cost function using incremental smoothing and mapping with the Bayes tree (iSAM2) [20].

B. *LiDAR factor Generation*

SSL-SLAM [2] is used for generating the LiDAR factors. The algorithm extracts feature points (planes/ edges)(III-B.1), and matches these to earlier found feature points that are stored in a local map. These correspondences are used to build up a cost function that is dependant on the relative movement. This cost function is minimized to estimate the relative movement (III-B.2).
The relative movement estimate is used to generate the factor for the sensor fusion (III-B.3).

1) Feature Extraction: Let the point cloud of a new frame k be written as $\mathcal{P}_k$. The point cloud is split into $M \times N$ cells based on the horizontal and vertical angle of the observed points. Here M is half of the total number of observed points in vertical direction. N is half of the total number of observed points in horizontal direction.
For every cell created by this procedure the local smoothness is calculated with (5). The local smoothness is high for edge features, while it is low for plane features[2].

$$\sigma_k^{m,n} = \frac{1}{\lambda^2} \cdot \sum_{\mathbf{p}_k^{(i,j)} \in \mathcal{S}_k^{(m,n)}} (||\mathbf{p}_k^{(i,j)}||_2^2 - ||\mathbf{p}_k^{(m,n)}||_2^2) \tag{5}$$

Here (m,n) are the cell numbers of the cells in $\mathcal{P}_k$. $\mathbf{p}_k^{(m,n)}$ is the geometry center of all points within the cell. The smoothness metric is formed by taking the difference between this geometry center and individual points in the cell as well as cells within a search radius λ.
The cells with the highest $\sigma_k^{(m,n)}$ are selected as edge features. The cells with the lowest $\sigma_k^{(m,n)}$ are selected as plane features.

2) Relative Movement: To find the relative movement between scan k and $k-1$, the feature points found in Sec. III-

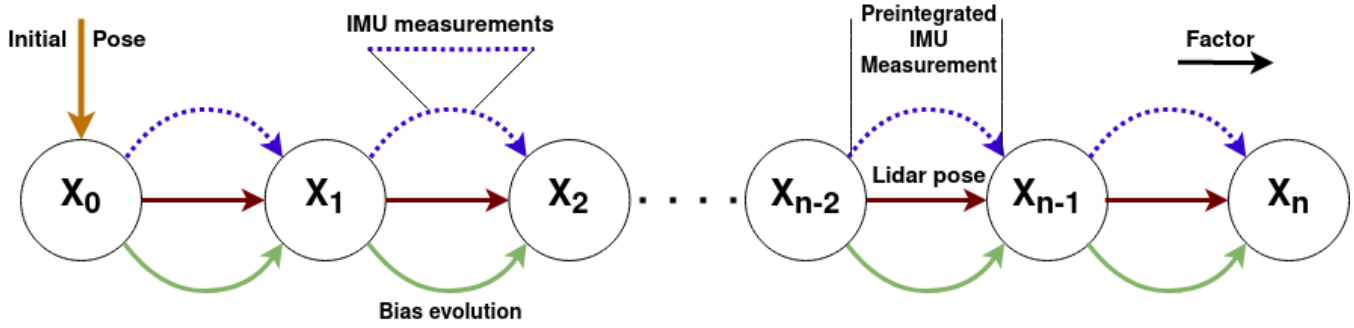


Fig. 3: The factor graph model of our method. The states $\mathbf{x}_0$:$\mathbf{x}_n$ are the unknowns to be optimized. The edges of the graph (the factors) are generated by preintegrated IMU measurements, LiDAR scans, the bias model and an initial pose.

B.1 are compared to a local map M_k generated by previous scans. The local map $M_k = \{\mathcal{P}_{k-1}, \mathcal{P}_{k-2}, ..., \mathcal{P}_{k-q}\}$ is defined by the previous q scans.

The edge and feature points found in k are transformed to the local map by applying homogenous transformation $\mathbf{T}_k \in SE(3)$ to find corresponding edge and feature points in previous scans. Now that the feature point from k is transformed to the local map, the closest corresponding feature point for each feature point is found. The local map is build with a K-D tree to improve search efficiency. When two matches are found for edge features, or three matches for plane features, they can be used to define the error metric that is used to find the relative LiDAR transformation.

The edge points use the point to line distance error metric to define the error. For every edge point $\mathbf{p}_k$ and it's correspondences $\mathbf{p}_1$,$\mathbf{p}_2$ the error is defined as [2]:

$$f_{\mathcal{E}}(\hat{\mathbf{p}}_k) = \frac{|(\hat{\mathbf{p}}_k - \mathbf{p}_2) \times (\hat{\mathbf{p}}_k - \mathbf{p}_1)|}{|(\mathbf{p}_2 - \mathbf{p}_1)|} \tag{6}$$

The plane points use the point to plane distance error metric to define the error. For every plane point $\mathbf{p}_k$ and its correspondences $\mathbf{p}_1$,$\mathbf{p}_2$,$\mathbf{p}_3$ the following error metric is used [2]:

$$f_{\mathcal{P}}(\hat{\mathbf{p}}_k) = |(\hat{\mathbf{p}}_k - \mathbf{p}_1)^T \cdot \frac{(\mathbf{p}_1 - \mathbf{p}_2) \times (\mathbf{p}_1 - \mathbf{p}_3)}{|(\mathbf{p}_1 - \mathbf{p}_2) \times (\mathbf{p}_1 - \mathbf{p}_3)|}| \tag{7}$$

The current position $\bar{\mathbf{T}}_k$ is found by stacking Equation 6 and 7 for every observed edge and plane feature and minimizing the cost (8) [2].

$$\bar{\mathbf{T}}_k \rightarrow \arg\min_{\mathbf{T_k}} \sum f_{\mathcal{E}}(\hat{\mathbf{p}}_k) + f_{\mathcal{P}}(\hat{\mathbf{p}}_k) \tag{8}$$

3) Factor Generation: The reprojection error $\mathbf{r}_L(x)$ is defined as the difference between the estimated relative transformation $\mathbf{T}_k - \mathbf{T}_{k-1}$ of the graph based algorithm and the measured relative transformation $\bar{\mathbf{T}}_k - \mathbf{T}_{k-1}$.

$$\mathbf{r}_L(k) = \bar{\mathbf{T}}_k - \mathbf{T}_{k-1} \tag{9}$$

The weight of the reprojection error for the LiDAR estimate is assessed by verifying whether the current measurement is an outlier. In the case that the difference between the IMU estimate and the LiDAR estimate exceeds a threshold, the LiDAR pose estimate is considered an outlier, and given a low weight. In the case that the LiDAR pose doesn't exceed the threshold, a high weight is passed.

The measurement uncertainty values passed to the algorithm for the LiDAR estimates have been determined empirically by minimizing the end-to-end trajectory error on the double staircase dataset (Fig. 6).

C. *Preintegrated Inertial Factor generation*

The measurements of the IMU are converted to preintegrated IMU measurements [25]. Preintegrated IMU measurements summarize multiple IMU measurements between two subsequent state updates into one single measurement [25]. This is done instead of updating the graph for every measurement, since the IMU update frequency (200 Hz) would make the graph unnecessarily large.

The gyroscope and the accelerometer measurements of the IMU are modelled as a perfect measurement with additional bias and Gaussian noise (10, 11).

$${}^B\boldsymbol{\omega}_m = {}^B\boldsymbol{\omega}_r + \mathbf{b}_g + {}^B\boldsymbol{\lambda}_\omega \tag{10}$$

$${}^B\boldsymbol{a}_m = {}^B\boldsymbol{a}_r + \mathbf{b}_a + {}^W\mathbf{g} + {}^B\boldsymbol{\lambda}_a \tag{11}$$

Here $\boldsymbol{\omega}_r$ and $\boldsymbol{a}_r$ are the real angular velocity and linear acceleration. They consist of respectively the measurements $\boldsymbol{\omega}_m$ and $\boldsymbol{a}_m$, the biases of the sensor $\mathbf{b}_a, \mathbf{b}_g$ and additional gaussian noise $\boldsymbol{\lambda}_{\boldsymbol{\omega}}, \boldsymbol{\lambda}_{\boldsymbol{a}}$. For the construction of the preintegrated IMU factor the bias is assumed to be known beforehand and constant between two LiDAR pose updates.

Changes in bias are later implemented in Eq. 21-23.
$\boldsymbol{\omega}_r$ and $\boldsymbol{a}_r$ can be integrated (twice in the case of a_r) to obtain the orientation and position of the sensor at the next timestep (12-14)[25].

$$^W R_B(i+1) = {}^W R_B(i)\text{Exp}({}^B\boldsymbol{\omega}_r(i)\Delta t) \tag{12}$$
$$^W \mathbf{v}(i+1) = {}^W \mathbf{v}(i) + {}^W \mathbf{a}_r(i)\Delta t \tag{13}$$
$$^W \mathbf{p}(i+1) = {}^W \mathbf{p}(i) + {}^W \mathbf{v}(i)\Delta t + \frac{1}{2} {}^W \mathbf{a}_r(i)\Delta t^2 \tag{14}$$

Here Δt is the time difference between two LiDAR scans. By substituting (10, 11) into (12-14), and transforming $^W\mathbf{a}_r(i)$ to the B frame, the equation relating the measurements to state updates is obtained (15-17).

$$^W R_B(i+1) = {}^W R_B(i)\text{Exp}((\boldsymbol{\omega}_m(i) - \mathbf{b}_g(i) - \boldsymbol{\lambda}_\omega(i))\Delta t) \tag{15}$$

$$\begin{aligned}\mathbf{v}(i+1) = \mathbf{v}(i) &+ \mathbf{g}\Delta t \\ &+ R(i)({}^B\mathbf{a}_m(i) - \mathbf{b}_a(i) - \boldsymbol{\lambda}_a(i))\Delta t\end{aligned} \tag{16}$$

$$\begin{aligned}\mathbf{p}(i+1) = \mathbf{p}(i) + \mathbf{v}(i) &+ \frac{1}{2}\mathbf{g}\ \Delta t^2 \\ &+ \frac{1}{2}\ {}^W R_B(i)({}^B\mathbf{a}_m(i) - \mathbf{b}_a(i) - \boldsymbol{\lambda}_a(i))\Delta t^2\end{aligned} \tag{17}$$

By iterating (15-17), and then compensating for the initial state (18-20) we finally get the relative movements $\Delta\mathbf{v}_{ij}, \Delta\mathbf{p}_{ij}, \Delta R_{ij}$, that describe the motion between the state at timestamp i and any arbitrary future state j.

$$\begin{aligned}\Delta\mathbf{p}_{ij} &= R_i^T\Big(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2\Big) \\ &= \sum_{k=i}^{j-1}\Big(\frac{3}{2}\Delta R_{ik}\big(\mathbf{a}_m(k) - \mathbf{b}_a(k) - \boldsymbol{\lambda}_a(k)\big)\Delta t^2\Big)\end{aligned} \tag{18}$$

$$\begin{aligned}\Delta\mathbf{v}_{ij} &= R_i^T\big(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}\big) \\ &= \sum_{k=i}^{j-1}\Delta R_{ik}\big(\mathbf{a}_m(k) - \mathbf{b}_a(k) - \boldsymbol{\lambda}_a(k)\big)\Delta t\end{aligned} \tag{19}$$

$$\begin{aligned}\Delta R_{ij} &= R_i^T R_j \\ &= \prod_{k=i}^{j-1}\text{Exp}\Big(\big(\omega_m(k) - \mathbf{b}_g(k) - \boldsymbol{\lambda}_\omega(k)\big)\Delta t\Big)\end{aligned} \tag{20}$$

The equations on the top lines of (18-20) are dependant on the nodes of the factor graph, the equations on the bottom lines of (18-20) only depend on the measurements, and are therefore independent of the states in the graph. The residuals errors are defined as the difference between the two lines of each equation (21-23).

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{p}_{ij}} = R_i^T\Big(&\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2\Big) \\ &- \Delta\mathbf{p}_{m,ij} - \delta\mathbf{p}_{ij}(b)\end{aligned} \tag{21}$$

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{v}_{ij}} = R_i^T\big(&\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}\big) \\ &- \Delta\mathbf{v}_{m,ij} + \delta\mathbf{v}_{ij}(b)\end{aligned} \tag{22}$$

$$\mathbf{r}_{\Delta R_{ij}} = \text{Log}\Big(\big(\Delta R_{m,ij}\text{Exp}(\delta R_{ij}(b))\big)^T R_i^T R_j\Big) \tag{23}$$

$$\mathbf{r}_I(j) = \begin{bmatrix}\mathbf{r}_{\Delta\mathbf{p}_{ij}} & \mathbf{r}_{\Delta R_{ij}} & \mathbf{r}_{\Delta\mathbf{v}_{ij}} & \mathbf{0}\end{bmatrix}^T \tag{24}$$

Here the δ's are added to incorporate changes in the bias. For a full definition of the δ's the reader is referred to [25]. By formulating the residual errors in this manner, only the small deviation δ's need to be updated when the bias changes. This avoids having to change the linearization point of the optimization, and having to reiterate through each individual IMU measurement again.

D. Bias Factor Generation

The bias forms a separate factor in the factor graph (Fig. 3). The evolution of the bias is assumed to be a zero mean Brownian motion [25] with variance η_{gd}, η_{ad} for the gyroscope and accelerometer respectively. The residual error between two subsequent points in time i and j can than be defined with the change in bias (25)[25].

$$\|\mathbf{r}_b(j)\|^2 = \|\mathbf{b}_j^g - \mathbf{b}_i^g\|_{\Sigma_g}^2 + \|\mathbf{b}_j^a - \mathbf{b}_i^a\|_{\Sigma_a}^2 \tag{25}$$

with:

$$\Sigma_g = \Delta t_{i,j}\text{Cov}(\eta_{gd}) \tag{26}$$
$$\Sigma_a = \Delta t_{i,j}\text{Cov}(\eta_{ad}) \tag{27}$$

Here $\mathbf{b}_i^g, \mathbf{b}_j^g$ is the gyroscopic bias at i and j. $\mathbf{b}_i^a, \mathbf{b}_j^a$ is the accelerometer bias at i and j. $\Delta t_{i,j}$ is the time difference between i and j.

E. Solving the Factor Graph

The graph consists of nodes and edges. The nodes (circles in Fig. 3) contain the state of the system at a single time step (2). The nodes are the unknowns in the system that need to be estimated. The edges, that are known, are the relations between the nodes. The factors created by the LiDAR and IMU are inserted as edges into the graph with the GTSAM [21] toolbox.

The values are initialized by the preintegrated IMU measurements. The nodes are then estimated incrementally using the Incremental Smoothing and Mapping algorithm (ISAM2) [20]. ISAM2 stores the variables of the factor graph in an efficient data structure that takes into account the effect that changing a variable has on other variables in the graph. When adding new variables to the graph, ISAM2 only updates the states that are affected by the new update to reduce computational cost. The full algorithm is summarized in Algorithm 1. A graph based model has been chosen because of the ability to implement loop closures [21] and because it can be updated incrementally allowing real time performance, even the graph becomes very large [20].

Algorithm 1 Factor Graph Optimization Node

Input: LiDAR scans $L_{0:n}$, IMU measurements $Z_{0:m}$
Prerequisites: Last LiDAR Keyframe L_i, IMU queue $Z_{i:k}$, Nonlinear Factor Graph $\mathbb{F}$
Output: Trajectory $\mathbf{x}_{1:n}$
begin
while Running **do**
  if IMU measurement Z_{k+1} **then**
    append Z_{k+1} to $Z_{i:k}$
  else if LiDAR scan L_j **then**
    Extract Edge and Plane features from L_j
    Compute Relative LiDAR transformation H_{ij}^{L}
    Compute Relative IMU transformation H_{ij}^{I}
    if abs$(H_{ij}^{L} - H_{ij}^{I}) <$ threshold **then**
      create factor $F_{L,ij}$ with small Σ_L
    else
      create factor $F_{L,ij}$ with high Σ_L
    Add $F_{L,ij}$ to Nonlinear factor graph $\mathbb{F}$
    Create IMU preintegration factor $F_{I,ij}$ from $Z_{i:k}$
    Add $F_{I,ij}$ to Nonlinear factor graph $\mathbb{F}$
    Create Bias Factor $F_{B,ij}$
    Add $F_{B,ij}$ to Nonlinear factor graph $\mathbb{F}$
    Solve $\mathbb{F}$ with ISAM2 to get $\mathbf{x}_{1:n}$
    update key from i to $i+1$
    Empty IMU queue $Z_{i:k}$
  end if
end if
end while
return $\mathbf{x}_{1:n}$, $\mathbb{M}$
end

IV. EXPERIMENTAL EVALUATION

The algorithm is tested in four different situations to evaluate the performance (Table I). In each of the experiments, the Intel L515 was held in hand by a person walking through the environment. The starting location was marked in each experiment, and the person walking with the sensor returned to the starting location at the end of the experiment. The experiments are designed to test the robustness and performance of the sensor to ambient (sun) light, vertical movement, and degenerate mapping scenarios. The Intel L515 datasets are captured and processed by a computer with 16gb RAM and AMD Ryzen 5 4500u processor running Ubuntu 20.04 LTS and ROS Noetic [27]. The bag files were recorded at 15 frames per second for the LiDAR scans and 200 Hz for the IMU.

TABLE I: Dataset Overview. At point insufficiencies $<$ 20 edge and/or plane points are available for the LiDAR estimation.

Dataset	Duration[s]	Point Insufficiencies	Challenge
Office (Night)	87.4	0	Optimal LiDAR conditions
Office (Day)	85.15	2	Mild Ambient Sun light
Staircase	230.8	0	Vertical Movement
Open Spaces	167	10	Degenerate scenarios

TABLE II: End-to-end translation error in meters (Z-translation error).

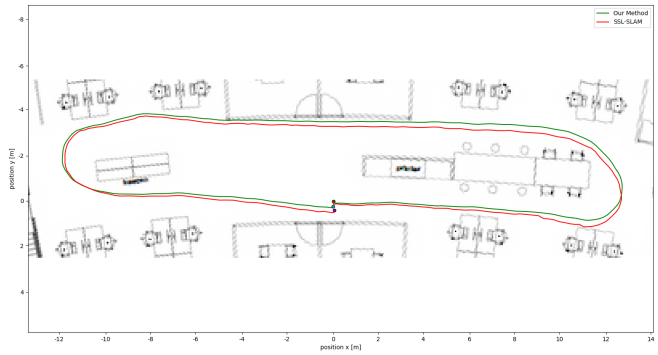
Dataset	SSL-SLAM	Our Method
Office (Night)	**1.27** (-1.20)	**0.31** (0.23)
Office (Day)	**0.825** (0.251)	**0.368** (0.268)
Staircase	**6.406** (4.581)	**2.362** (0.563)
Open Spaces	$>$**10**	$>$**10**

A. The Office (Night)

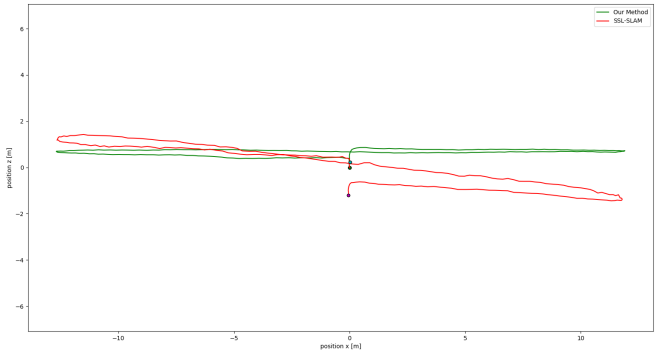
The Office (Night) experiment is conducted in a feature rich office environment. The office has got windows through which ambient (sun)light can enter the environment. This first experiment is done at night with the office lighting turned off to minimize the effect of ambient light on the sensor. Over the trajectory the proposed method achieved a lower end to end trajectory error (Fig. 4). The method especially reduces altitude drift (Table II).

B. The Office (Day)

The Office (Day) experiment is performed by walking along approximately the same trajectory in the same office environment as IV-A, but this time during the day. Ambient light enters the office through the windows that are located



(a) Trajectory comparison (x,y)



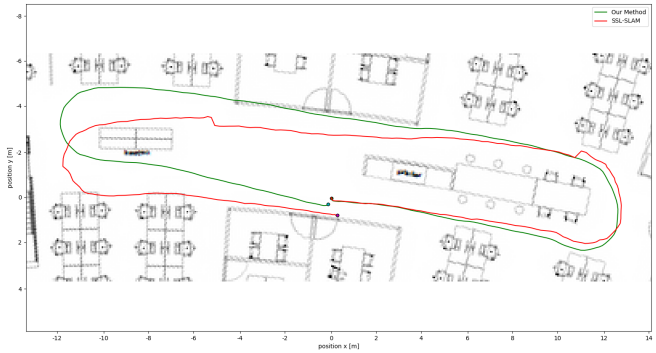
(b) Trajectory comparison (x,z)

Fig. 4: Results of walking through the office and returning at the starting position at night. The trajectory direction is counter clockwise

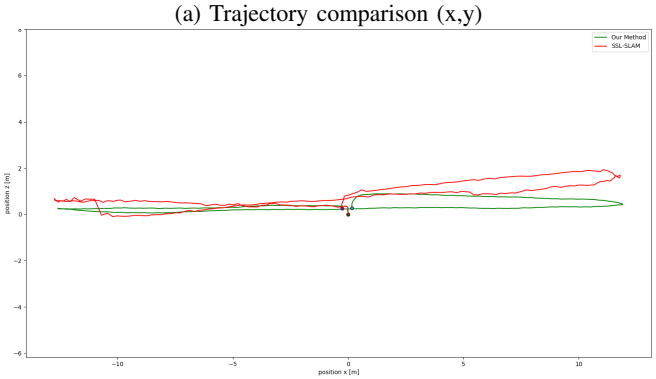
at the sides of the environment. The outside conditions were cloudy during the experiment: direct sunlight was absent and only indirect sunlight was present. This experiment is conducted to evaluate the effect of ambient sunlight on the sensor. During the experiment it occurred twice that there were few (less than 20) edge/plane feature points available for LiDAR localization. Our method achieved an end-to-end translation error of 0.368 m compared to 0.825 m for SSL-SLAM over the trajectory. During the experiment the altitude of the sensor was kept constant. Even though the end-to-end error in Z-direction is slightly less for SSL-SLAM (Fig. 5b) compared to our method, SSL-SLAM drifts more over the trajectory in Z-direction than our method. Our method performs comparable to the Office (Dark) experiment, while for SSL-SLAM the trajectory is at times more than 2 meters different.

C. *Staircase*

The Staircase dataset features two staircases to test vertical movement. In the experiment, the person holding the sensor walked toward a staircase and then climbed two floors. The person then turned around and descended the same stairs and eventually returned to the starting point of the experiment. Our method achieves an altitude error at the end of the trajectory of 0.56m (compared to 4.58m for SSL-SLAM). The end-to-end translation error is 2.36m (compared to 6.41m for SSL-SLAM). The trajectory is plotted in Fig. 6). The end-to-end translation error of our method is especially lowered in Z-direction as can be seen in Fig. 6b. The structure of the staircase can easily be seen when the trajectory is plotted in 3D (figure 6c).

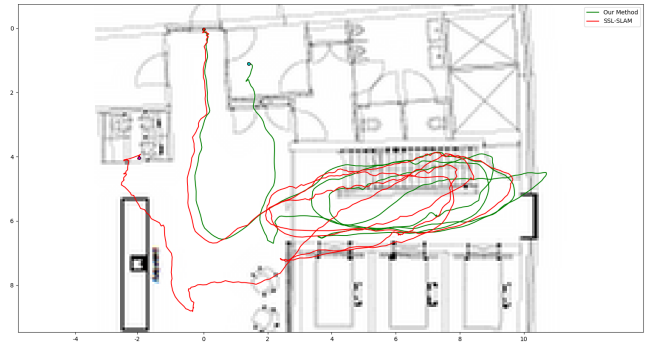


(a) Trajectory comparison (x,y)

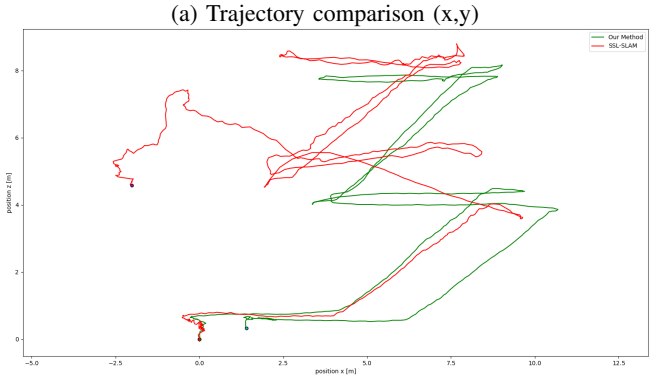


(b) Trajectory comparison (x,z)

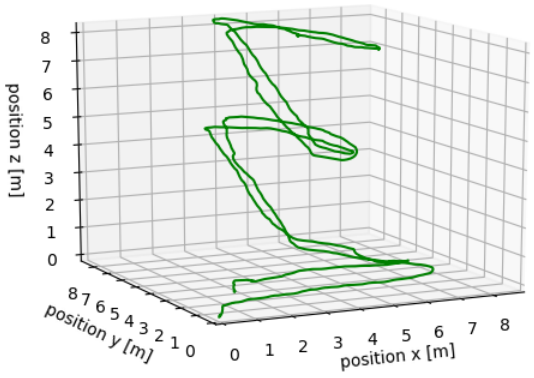
Fig. 5: Results of walking through the office and returning at the starting position during the day. The trajectory direction is counter clockwise.



(a) Trajectory comparison (x,y)



(b) Trajectory comparison (x,z)



(c) 3D Trajectory of our method

Fig. 6: Results of the Staircase experiment.

D. Open Spaces

The Open Spaces experiment was done by traversing the trajectory in Fig. 7. The parking garage features large open spaces, were features are scarce and often more than 5 m away from the sensor. The experiment was done at night to minimize the influence of ambient sunlight on the performance. This experiment was done to test the performance of the method in situations were features are far away or mostly on the same plane (in case of the roof of the parking garage). Both our method and SSL-SLAM failed to generate meaningful results (Table II). The optimization of III-E eventually became ill constrained in the case of our method. For SSL-SLAM the trajectory diverged multiple meters from the real trajectory within seconds.

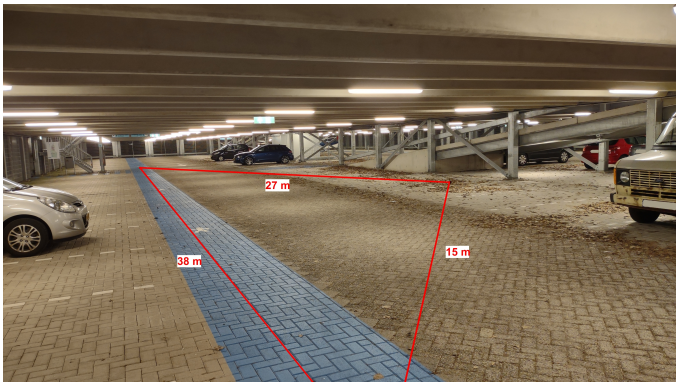


Fig. 7: During the Open Spaces experiment the red triangle was traversed in counter clockwise direction. Features are mostly located on the beams of the roof and on the floor.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this work we presented a tightly coupled graph based method for fusing LiDAR and IMU measurements for the Intel L515 LiDAR Sensor. The IMU factors in the graph are generated by using IMU preintegration [25]. The LiDAR factors are generated through feature based scan matching based on SSL-SLAM [2]. These factors are combined in a non-linear factor graph to estimate the pose of the sensor. Solving the factor graph is done incrementally with ISAM2 [20] in the GTSAM toolbox [21].

Our method results in reduced drift compared to SSL-SLAM and especially reduces altitude drift. The algorithm was tested in various environments, with different light conditions, vertical movement and feature availability. The algorithm is robust against short failures in the LiDAR estimation, but fails in the case of of longer failures when not enough features are present.

B. Future Works

Currently the algorithm suffers from long term drift. Long term drift in graph based algorithms can be reduced by creating edges between non sequential states in the factor graph (figure 3). These loop closures have been successfully

implemented in visual algorithms[13][31], as well as LiDAR algorithms [10] to reduce long term drift.

The algorithm can also be improved by using the camera of the Intel L515 to perform visual odometry. Visual odometry approaches like [13] [4] [5] have shown robust performance in outdoor conditions where ambient (non-direct) sunlight is present, like the KITTI [39] dataset. Visual odometry can particularly help in cases where the L515 LiDAR input is handicapped by ambient (non-direct) sunlight, or where there are many visual features but not enough LiDAR features are available, like in Section IV-D.

VI. ACKNOWLEDGMENTS

SSL-SLAM-code [41] was used for creating the LiDAR factors. GTSAM-fusion [40] code was used for the GTSAM [21] implementation.

REFERENCES

[1] Zhang, Ji & Singh, Sanjiv. (2017). Low-drift and Real-time Lidar Odometry and Mapping. Autonomous Robots. 41. 401-416.

[2] H. Wang, C. Wang and L. Xie, "Lightweight 3-D Localization and Mapping for Solid-State LiDAR," in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1801-1807, April 2021, doi: 10.1109/LRA.2021.3060392.

[3] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 2174-2181

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in Proc. IEEE Intl. Conf. on Robotics and Automation, Hong Kong, China, June 2014, pp. 15–22.

[5] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monoc- ular visual-inertial state estimator," IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004–1020, 2018

[6] P.J. Besl and N.D. McKay, "A Method for Registration of 3D Shapes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14(2): 239-256, 1992.

[7] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 3126-3131, doi: 10.1109/ICRA40945.2020.9197440.

[8] Shan, Tixiao & Englot, Brendan & Ratti, Carlo & Rus, Daniela. (2021). LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping.

[9] Cvišić, I, Ćesić, J, Marković, I, Petrović, I. SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles. J Field Robotics. 2018; 35: 578– 595.

[10] Shan, Tixiao et al. "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2020): 5135-5142.

[11] Zuo, Xingxing et al. "LIC-Fusion 2.0: LiDAR-Inertial-Camera Odometry with Sliding-Window Plane-Feature Tracking." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2020): 5112-5119.

[12] X. Zuo, P. Geneva, W. Lee, Y. Liu and G. Huang, "LIC-Fusion: LiDAR-Inertial-Camera Odometry," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 5848-5854, doi: 10.1109/IROS40897.2019.8967746.

[13] Mur-Artal, Raul & Montiel, J. & Tardos, Juan. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Transactions on Robotics. 31. 1147 - 1163. 10.1109/TRO.2015.2463671.

[14] Mur-Artal, Raul & Tardos, Juan. (2016). ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. IEEE Transactions on Robotics. PP. 10.1109/TRO.2017.2705103.

[15] Campos, Carlos et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM." IEEE Transactions on Robotics 37 (2021): 1874-1890.

[16] M. Labbé and F. Michaud, "RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation," in Journal of Field Robotics, vol. 36, no. 2, pp. 416–446, 2019.

[17] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pp. 1052–1067, 2007.

[18] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground- optimized Lidar Odometry and Mapping on Variable Terrain," IEEE/RSJ International Conference on Intelligent Robots and Systems,

[19] Myriam Servieres, Valérie Renaudin, Alexis Dupuis, Nicolas Antigny. Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking. Journal of Sensors, 2021, 2021, 26 p

[20] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 3281-3288, doi: 10.1109/ICRA.2011.5979641.

[21] Frank Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction", September 2012, Technical Report number GT-RIM-CP&R-2012-002

[22] Diehl S, Neuvel JM, Zlatanova S, Scholten H (2006) Investigation of user requirements in the emergency response sector: the Dutch case. In: van de Walle B, Song Y, Zlatanova S, Li J (eds) Second symposium on Gi4DM, Goa, India, p 6

[23] Ferreira, A. F. G., Fernandes, D. M. A., Catarino, A. P.; Monteiro, J. L. (2017). Localization and Positioning Systems for Emergency Responders: A Survey. IEEE Communications Surveys and Tutorials, 19(4), 2836–2870. https://doi.org/10.1109/COMST.2017.2703620

[24] M. Demir and K. Fujimura, "Robust Localization with Low-Mounted Multiple LiDARs in Urban Environments," IEEE Intelligent Transportation Systems Conference, pp. 3288-3293, 2019

[25] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," IEEE Transactions on Robotics, vol. 33(1): 1-21, 2016.

[26] T. Lupton and S. Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. IEEE Trans. Robotics, 28(1): 61–76, Feb 2012.

[27] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng, "ROS: An Open-source Robot Operating System," IEEE ICRA Workshop on Open Source Software, 2009.

[28] Intel® RealSense™ LiDAR Camera L515 Datasheet, Revision 003, January 2021

[29] Kschischang, F., Frey, B., and Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. IEEE Trans. Inf. Theory, 47(2)

[30] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," in IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99-110, June 2006, doi: 10.1109/MRA.2006.1638022.

[31] Labbé, M, Michaud, F. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. J Field Robotics. 2019; 35: 416– 446.

[32] Kudriashov, Andrii and Buratowski, Tomasz and Giergiel, Mariusz and Małka, Piotr,"SLAM as Probabilistic Robotics Framework Approach" in "SLAM Techniques Application for Mobile Robot in Rough Terrain", 2020, "39–64"

[33] He, Ying et al. "An Iterative Closest Points Algorithm for Registration of 3D Laser Scanner Point Clouds with Geometric Fea-

tures." Sensors (Basel, Switzerland) vol. 17,8 1862. 11 Aug. 2017, doi:10.3390/s17081862

[34] S. Lynen, M.W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A Robust and Modular Multi-sensor Fusion Approach Applied to MAV Navigation," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3923-3929, 2013.

[35] S. Yang, X. Zhu, X. Nian, L. Feng, X. Qu, and T. Mal, "A Robust Pose Graph Approach for City Scale LiDAR Mapping," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1175-1182, 2018.

[36] M. Demir and K. Fujimura, "Robust Localization with Low-Mounted Multiple LiDARs in Urban Environments," IEEE Intelligent Transportation Systems Conference, pp. 3288-3293, 2019.

[37] Y. Gao, S. Liu, M. Atia, and A. Noureldin, "INS/GPS/LiDAR Integrated Navigation System for Urban and Indoor Environments using Hybrid Scan Matching Algorithm," Sensors, vol. 15(9): 23286-23302, 2015

[38] S. Hening, C.A. Ippolito, K.S. Krishnakumar, V. Stepanyan, and M. Teodorescu, "3D LiDAR SLAM integration with GPS/INS for UAVs in urban GPS-degraded environments," AIAA Infotech@Aerospace Conference, pp. 448-457, 2017

[39] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1231–1237, 2013

[40] P. Kemppi, "GTSAM-Fusion Github", `https://github.com/PaulKemppi/gtsam_fusion`, September 2020

[41] W. Hang, "ssl_slam Github", `https://github.com/wh200720041/ssl_slam`, Mar 2021

Chapter 4

Conclusions and Future Work

4-1 Conclusions

New advances in Micro-Electro-Mechanical System (MEMS) technology have allowed for the development of a cheaper and smaller kind of Light Detection and Ranging (LiDAR) sensor. These new sensors have a lower observation range than conventional LiDAR sensors, as well as decreased performance in situations that are rich in ambient sunlight.
In this thesis a new graph based lidar inertial localization method was developed and tested on the Intel® RealSense™ LiDAR Camera L515 (Intel L515) MEMS based LiDAR. The method was proposed to improve the localization accuracy of LiDAR Simultaneous Localization And Mapping (SLAM) algorithms on small low power LiDARs.
The graph based algorithm combines the measurements of the Inertial Measurement Unit (IMU) and LiDAR of the sensor to determine the location, orientation and IMU bias of the sensor.
These variables are determined in an optimization based approach that minimizes the sum of all the reprojection errors of the sensor measurements.
The sum of all the reprojection errors is generated by LiDAR pose estimates calculated by the SSL-SLAM algorithm [20], IMU preintegration measurements and a model for the evaluation of the IMU bias [5].
The algorithm is tested in various conditions that either focus on optimal performance or on one of the main situations in which localization is hard. The algorithm is compared to the performance of the SSL-SLAM algorithm [20], since this is currently the only algorithm tested on the Intel L515. The results are as follows:
Firstly, localization drift in general is reduced by applying the algorithm. Especially drift in direction of gravity is reduced. The algorithm was tested in dark conditions to reduce the effect of ambient light on the sensor. By testing the sensor in dark conditions an overall drift reduction of 75 % was achieved compared to Solid State Lidar Simultaneous Localization and Mapping (SSL-SLAM).
Secondly, the algorithm is able to reduce the effect of ambient sunlight on the performance when doing LiDAR SLAM in an office environment with ambient sunlight entering through the

windows. Thirdly, The algorithm was tested in degenerate situations. The outlier detection approach that compares the SSL-SLAM estimates with the IMU measurements is able to reduce the effect of short dropouts of accurate LiDAR localization due to degeneracy. These dropouts should not be longer than a few seconds, because otherwise the algorithm may lose track altogether. A review of this problem is presented in the future works.
From this thesis it can be concluded, that the localization accuracy of SSL-SLAM can be improved by fusing it in a graph based approach with IMU data.

4-2 Future Works

The algorithm currently is only able to withstand short outliers of the SSL-SLAM algorithm and only works partially in situations with ambient sunlight. Three approaches are recommended to counter longer stretches of outliers and handle scenes where features are not always available within the LiDAR range.

Implementing Visual Information

The system can be improved by incorporating the camera into the localization process. Visual odometry algorithms like Semi Direct Visual Odometry [6] and Orientated FAST Rotated BRIEF-SLAM (ORB-SLAM) [14] [15], [2] are proven to have good performance in outside conditions like the KITTI [7] dataset.
By adding a vision factor to the graph based localization algorithm, the algorithm is more robust against LiDAR dropout/degeneracies. Having full LiDAR visual inertial fusion counters the inability of visual only localization algorithms to work in the dark. also counters one of the main drawbacks of using a single camera for localization, which is the unobservability of scale.

Implementing Loop Closures

In LiDAR SLAM various methods for loop closures have been proposed [17]. Using a loop closure method allows the camera to re-localize after localization is lost. This allows the creation of non sequential factors in the factor graph.
The camera of the Intel L515 can also be used for implementing loop closures. By saving observed features found with the camera they can be stored. If they are later observed again, a feature matching algorithm can be used to create loop closures in the graph based algorithm. In the case of visual methods this has shown to reduce long term drift significantly [2].

Extending the Field of View

The Field of View (FOV) of the sensor is a cone with a 70 x 55 degrees viewing angle. The FOV can be increased by adding more sensors in order to come closer to a 360 degrees FOV, while still being significantly cheaper than a conventional LiDAR setup. It is expected that by extending the FOV with more sensors, the problem of scene degeneracy will be lessened up to a point. In situations where the sensors are over saturated with ambient sunlight that

leads to a blackout in localization, using another kind of sensor would be preferred over using multiple Intel L515's.

Bibliography

[1] Paul J Besl and Neil D McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[2] Carlos Campos, Richard Elvira, Juan J.Gomez Rodriguez, Jose M. Jose, and Juan D. Tardos. ORB-SLAM3: An Accurate Open-Source Library for Visual, Inertial, and Multimap SLAM. *IEEE Transactions on Robotics*, pages 1–15, 2021.

[3] Visual Clarity and Rapid Decisions. GEO4I ' S ANALYSIS OF 15 CM HD MAXAR, 2021.

[4] Hugh Durrant-Whyte and Tim Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Robotics and Automation Magazine*, 13, 2006.

[5] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. *Robotics: Science and Systems*, 11, 2015.

[6] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry. *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[7] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, (October):1–6, 2013.

[8] William Arthur Heidel. Anaximander's Book , the Earliest Known Geographical Treatise. *Proceedings of the American Academy of Arts and Sciences*, 56(7):239–288, 1921.

[9] Feng Huang, Weisong Wen, Jiachen Zhang, and Li-Ta Hsu. Point wise or Feature wise? Benchmark Comparison of Public Available LiDAR Odometry Algorithms in Urban Canyons. *Arxiv*, abs/2104.0, 2021.

[10] Intel. Intel ® RealSense TM LiDAR Camera L515 Datasheet. (January), 2021.

[11] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. *Special Issue on the Ninth International Workshop on Algorithmic Foundations of Robotics (WAFR)*, 31(2):216–235, 2012.

[12] John Ryan Kidd. Performance Evaluation of the Velodyne VLP-16 System for Surface Feature Surveying. *Master's Theses and Capstones*, 2017.

[13] Manon Kok and Thomas B Schön. Estimation Using Inertial Sensors. *IEEE Signal Processing Letters*, 26(11):1673–1677, 2019.

[14] J M M Montiel, Raul Mur-Artal, and Juan D. Tardos. ORB-SLAM : A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[15] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[16] Tixiao Shan and Brendan Englot. LeGO-LOAM : Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (October):4758–4765, 2018.

[17] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LIO-SAM : Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (July):5135–5142, 2020.

[18] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping. *IEEE International Conference on Intelligent Robots and Systems*, pages 5135–5142, 2021.

[19] Dingkang Wang, Connor Watkins, and Huikai Xie. MEMS mirrors for LiDAR: A review. *Micromachines*, 11(5), may 2020.

[20] Han Wang, Chen Wang, and Lihua Xie. Lightweight 3-D Localization and Mapping for Solid-State LiDAR. *IEEE Robotics and Automation Letters*, (January):1–8, 2021.

[21] Wei Xu and Fu Zhang. FAST-LIO : A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021.

[22] Ji Zhang and Sanjiv Singh. LOAM: Lidar Odometry and mapping in Real-time. *Robotics: Science and Systems*, 2014.

[23] Xingxing Zuo, Patrick Geneva, Woosik Lee, Yong Liu, and Guoquan Huang. LIC-Fusion: LiDAR-Inertial-Camera Odometry. *IEEE International Conference on Intelligent Robots and Systems*, (September):5848–5854, 2019.

Glossary

List of Acronyms

FOV Field of View

LiDAR Light Detection and Ranging

ICP Iterative Closest Points

IMU Inertial Measurement Unit

Intel L515 Intel® RealSense™ LiDAR Camera L515

LIO-SAM Lidar Inertial Odometry via Smoothing And Mapping

LOAM LiDAR Odometry And Mapping

MEMS Micro-Electro-Mechanical System

MAP Maximum A Posteriori

ORB-SLAM Orientated FAST Rotated BRIEF-SLAM

SLAM Simultaneous Localization And Mapping

SNR signal-to-noise ratio

SSL-SLAM Solid State Lidar Simultaneous Localization and Mapping