Loosely coupled LiDAR-Visual/Thermal-Inertial Odometry and Mapping

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science & Engineering

by

Nikhil Vijay Khedekar

Prof. Dr. Kostas Alexis - Thesis Advisor
December 2020
THE GRADUATE SCHOOL

We recommend that the thesis prepared under our supervision by

Nikhil Vijay Khedekar

entitled

Loosely coupled LiDAR-Visual/Thermal-Inertial Odometry and Mapping

be accepted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Kostas Alexis, Ph.D.
Advisor

Christos Papachristos, Ph.D.
Committee Member

Davide Scaramuzza, Ph.D.
Committee Member

Anna Panorska, Ph.D.
Graduate School Representative

David W. Zeh, Ph.D., Dean
Graduate School

December, 2020
Abstract

This thesis presents a loosely coupled LiDAR-Visual/Thermal-Inertial odometry and mapping method that uses factor graphs. Our approach jointly optimizes relative pose constraints provided by a LiDAR scan-to-scan alignment method and a Visual/Thermal-Inertial method with preintegrated IMU constraints. An optimized relative pose prior is provided to a LiDAR scan-to-map alignment method to finally output the odometry of the system as well as globally registered pointclouds. A set of evaluation studies is presented showing the outperformance of our approach against the LiDAR only odometry and mapping method in a tunnel environment and the field-verification in an autonomous mission in an underground mine.
Dedication

Dedicated to my family.
Acknowledgments

First, I would like to thank my advisor, Dr. Kostas Alexis for his guidance and mentorship without which this work would not have been possible.

I would like to thank my thesis committee - Dr. Anna Panorska, Dr. Christos Papachristos and Dr. Davide Scaramuzza, for their valuable time, inputs and guidance that have helped me to complete this work.

I would also like to thank Dr. Shehryar Khattak for all of his help in this ongoing journey into state estimation. I would also like to thank all the members of the Autonomous Robots Lab, past and present, for their help, guidance and insightful discussions - Dr. Christos Papachristos, Dr. Tung Dang, Frank Mascarich, Huan Nguyen, Paolo De Petris, Mihir Dharmadhikari, Mihir Kulkarni, Russell Reinhart, Maria Tsiourva and Harpreet Singh. Finally, I would like to thank my family, for all their undying love and support.
## List of Figures

1.1 Field deployment of the Charlie robot in an underground mine visualizing the sensor-degraded environment (Left) and the map created by the proposed approach (Right). .................................................. 2

3.1 Architecture of LiDAR-Visual/Thermal-Inertial Odometry and Mapping. The thin arrows indicate the transfer of messages in the system while the thick arrow indicates the system output. .......................... 16

3.2 An instance of the generated factor graph after processing the point-cloud $P_{k=4}$ with the state variables (Purple), the prior factors (Yellow), Combined IMU factors (Red) and two types of between factors - from LiDAR scan-to-scan alignment (Green) and Depth-enhanced ROVIO/ROTIO (Blue) ................................................................. 17

4.1 The M100 Charlie platform utilized in the presented experiments. .......................... 23

4.2 This figure shows a comparison of the odometry and mapping result on using LiDAR only (LOAM) and LiDAR combined with visual and inertial terms (Proposed) in a tunnel in Northern Nevada, USA. The odometry is visualized in red in both images. .......................... 24
4.3 The top row of this figure shows the LiDAR-Thermal-Inertial odometry and mapping result in a deployment of the "Charlie" robot in an active underground gold mine in Northern Nevada, USA. Charlie is tasked to explore the environment using [1] and is later commanded to return-to-home. The accuracy of the approach is verified by virtue of the fact that the robot returns to the take-off spot to land. The middle and bottom rows illustrate the tracked Thermal-Inertial features and the environmental conditions at the specific labelled instances in the mission.
Chapter 1

Introduction and Related Work

The ability of robotic systems to operate autonomously hinges upon their capacity to estimate their pose in the environment and reconstruct a map of their surroundings. Especially in GPS-denied environments, but also more broadly, this capacity depends upon the set of technologies that are collectively called Simultaneous Localization And Mapping (SLAM). SLAM is therefore a vital capability for robotic systems. The last 30 years of relevant research have resulted in pioneering SLAM solutions capable of long-term operation [2], reconstruction of accurate and dense maps [3] and precise localization in real-time [4]. However, the vast majority of research efforts in the domain have focused on the fusion of a single exteroceptive modality, such as LiDAR or visual cameras, possibly alongside inertial sensor cues [4–11]. Nevertheless, a set of contributions, such as the efforts in [12–15] have demonstrated the possible advantages of multi-modal fusion techniques. While vision-only systems struggle with darkness, obscurants, over-exposure and broadly lack of visual features, LiDAR-based solutions do not depend on visual degradation. Similarly, while LiDAR-only methods suffer in scenes that are self-similar (e.g., a symmetric straight tunnel), vision-based solutions are invariant to such cases of geometric ill-conditioning. It is thus only natural that the possible intelligent combination of LiDAR and visual modalities,
alongside Inertial Measurement Unit (IMU) cues can offer a more precise, robust and resourceful localization and mapping system.

![Field deployment of the Charlie robot in an underground mine visualizing the sensor-degraded environment (Left) and the map created by the proposed approach (Right).](image)

**Figure 1.1:** Field deployment of the Charlie robot in an underground mine visualizing the sensor-degraded environment (Left) and the map created by the proposed approach (Right).

SLAM systems are organized in a set of different operational principles, typically with respect to the way sequentially acquired data are associated and registered to each other, as well as how front-end provided measurements are (co-)optimized to infer the robot pose and the map of its environment. With respect to the latter, the so-called SLAM back-end, the literature is generally organized into two large groups of methods, namely filtering-based and batch optimization-based methods. EKF-based methods have been the traditional choice ever since Smith, Self and Cheeseman presented EKF-SLAM [16], one of the pioneering solutions relying on the Extended Kalman Filter (EKF). The method is computationally efficient, in most cases robust, and indeed can accommodate the fusion of multiple modalities as individual updates in the EKF Markov chain. Nevertheless, its efficiency comes at the cost of certain inconsistencies introduced due to the inherent non-linearity of the SLAM problem, which for the EKF-SLAM solution, gets linearized at every time-step and marginalized into the filter. Batch optimization methods alleviate this problem by optimizing over the entire state and maintaining a richer set of frame-to-frame data (landmark) associations. This strategy naturally comes at an increased computational cost but
at the same time allows for a more accurate and more resilient estimation process. Batch optimization can be also be performed online in a sliding window fashion. Intelligent algorithm designs, such as those found in [6,8,10], provide efficient ways to select the most informative recent observations and maintain a robust optimization problem that remains computationally tractable. This in combination with recent advances in hardware and software libraries enables the use of such methods more widely.

This work contributes a loosely-coupled method for the fusion of LiDAR and visual or thermal cameras, combined with IMU data. Two sensor front-ends, namely for LiDAR-based and visual/thermal-based iterative estimation of the evolving robot pose transformations are utilized to incorporate optimization constraints in a smoothing solution responsible for the optimization back-end tasks. The latter further fuses pre-integrated IMU constraints based on the work in [17]. To formulate the problem we utilize the concept of factor graphs [18] which are well-suited to modeling complex estimation problems such as SLAM. A factor graph is a bipartite graph that consists of factors connected to variables. The variables represent the unknown random variables in the estimation problem, whereas the factors represent probabilistic constraints on those variables. For our problem, relative visual/thermal and LiDAR constraints are modeled as ”between factors”. The GTSAM toolbox (“Georgia Tech Smoothing and Mapping) [19] is utilized to formulate and solve this problem efficiently as it provides robust and efficient algorithms for iterative estimation.

The motivation behind this work relates to the deployment of autonomous robotic systems in subterranean environments and other settings subject to sensor-degradation. Previous work of our team has presented a set of contributions towards enabling resilient localization in environments that present self-similar geometry, dense obscurants, darkness and other cases of degradation [15,20,21]. With this contribution we
take one step forward towards more versatile, more extendable and more robust multi-
modal localization and mapping, deployable across a set of robotic configurations, be
it aerial, legged or roving systems.

The remainder of this thesis is organized as follows: Chapter 2 presents the back-
ground and theory followed by the approach in Chapter 3. The evaluation studies
conducted are described in Chapter 4 while the conclusions and future work are de-
scribed in Chapter 5.
Chapter 2

Background

2.1 Base Methods

We first briefly describe the base methods of LOAM and Depth-enhanced ROVIO/ROTIO.

2.1.1 LOAM

LiDAR Odometry and Mapping in Real-Time was originally proposed in [4] as a real-time method to provide highly accurate odometry and reconstruct a pointcloud based map of the environment using a rotating 2-axis LiDAR. The key contribution of the method is a high-frequency low-fidelity odometry estimator for velocity estimation coupled with a low-frequency fine matching and registration of pointclouds for mapping. Similar to other LiDAR based methods, both these steps essentially try to minimize the error between extracted correspondences.

LiDAR Odometry

Feature extraction: Given a pointcloud, $P_k$, feature points are extracted from it and classified as coming from edges or planar patches. The classification is done on
the basis of the “smoothness” of the local surface around the point. Suppose a point \( i \) in \( P_k, i \in P_k \), and the set of consecutive points in the scan line containing \( i \) as \( S \), then the smoothness is defined by

\[
c = \frac{1}{|S| \cdot \|X^L_{(k,i)}\|} \left\| \sum_{j \in S, j \neq i} (X^L_{(k,i)} - X^L_{(k,j)}) \right\| \quad (2.1)
\]

where \( X \) denotes the 3D position of a LiDAR point. Points in a scan line are sorted on the basis of their \( c \) values and classified with high \( c \) values corresponding to edges and low \( c \) values corresponding to planar patches. Furthermore, the scan lines are divided into subregions with a maximum number of possible edge or planar points to allow for a well-distributed set of feature points. The extracted feature points from \( P_k \) form a set of edge points \( E_k \) and a set of planar points \( H_k \).

**Scan-to-Scan Alignment:** Given two consecutive pointclouds, \( P_k \) and \( P_{k+1} \), the features sets \( E_k, H_k \) and \( E_{k+1} \) and \( H_{k+1} \) are extracted and the correspondences are found. The correspondence for an edge point is a line described by 2 edge points in the previous scan whereas for a planar point is a planar patch described by 3 points in the previous scan. The overall distance between correspondences is then minimized to estimate the scan-to-scan motion. For an edge point, \( i \in E_{k+1} \), the distance to the corresponding line given by \( (j, l) \), \( j, l \in P_k \) is found using

\[
d_E = \left| \frac{(X^L_{(k+1,i)} - X^L_{(k,j)}) \times (X^L_{(k,i)} - X^L_{(k,l)})}{|X^L_{(k,j)} - X^L_{(k,l)}|} \right| \quad (2.2)
\]

where \( X^L_{(k+1,i)}, X^L_{(k,j)} \) and \( X^L_{(k,l)} \) are the coordinates of the points \( i, j \) and \( l \) in the frame \( L_{k+1} \), respectively. For a planar point, \( i \in H_{k+1} \), the distance to the corresponding planar patch given by \( j, l, m \in P_k \), is
\[ d_H = \frac{\left( X^L_{(k+1,i)} - X^L_{(k,j)} \right) \cdot (X^L_{(k,j)} - X^L_{(k,l)}) \times (X^L_{(k,j)} - X^L_{(k,m)})}{\left| (X^L_{(k,j)} - X^L_{(k,l)}) \times (X^L_{(k,j)} - X^L_{(k,m)}) \right|} \]  

(2.3)

where \( X^L_{(k,m)} \) are the coordinates of point \( m \) in the frame \( L_{k+1} \). These distances for the selected feature points are all jointly optimized in a robust manner using the Levenberg-Marquardt algorithm to provide the relative transform \( T^L_{k+1} \) from the frame \( L_{k+1} \) to \( L_k \) expressed in \( L_{k+1} \).

**LiDAR Mapping**

The Mapping algorithm runs at 1Hz and further optimizes the transform in similar manner as in the previous subsection but with 10 times the number of feature points. This optimization refines the transformation and is fused with the result of the odometry step to provide an odometry update as well as a registered pointcloud in real time.

As the overall method optimizes over the distances between a set of feature correspondences, the lack of sufficient features, as in the case of geometrically self-similar environments, can cause the solution to be degenerate. Further details on the method can be found in [4,13].

### 2.1.2 Depth-Enhanced ROVIO/ROTIO

**ROVIO**

Robust Visual Inertial Odometry (ROVIO) is a monocular Visual Inertial Odometry (VIO) approach that is formulated by an Extended Kalman Filter (EKF) that tracks a small set of multi-level patch features as part of the state. The propogation in the
EKF is carried out using acceleration, $\mathbf{a}$, and angular velocity values, $\omega$, provided by the IMU while the update step is carried out using the photometric error for the tracked patches. The method parametrizes the 3D locations of the landmarks using a robot-centric approach (i.e. their locations are always with respect to the current camera pose) with each landmark being represented using a 2D bearing vector, $\mu$, parametrized by the azimuth and elevation angles and $\rho$, the inverse depth of the feature [22]. The method can also estimate the camera-IMU extrinsics online with these being part of the state. New candidate features are detected using the FAST corner detector [23] and selected based on their adapted Shi-Tomasi scores which essentially considers the combined hessian on multiple levels of the image. The method also implements a global and local quality score that is used to evaluate which features are to be added/kept in the filter state. $\mathbf{a}$ and $\omega$ are used for predicting the new locations of the tracked features in each step. Maintaining three coordinate frames, $I$ for the IMU frame, $C$ for the camera frame and $W$ for the world coordinate frame, the filter state is given by

$$
\mathbf{x} = (r, v, q, b_a, b_\omega, c, z, \mu_0, \ldots, \mu_N, \rho_0, \ldots, \rho_N) \quad (2.4)
$$

where $r$, $v$ are the position and velocity of the robot expressed in the $I$ coordinate frame, respectively, $q$ represents the IMU attitude from the $I$ to $W$ coordinate frame, $b_a$ and $b_\omega$ represent the additive accelerometer and gyroscope biases respectively, expressed in the $I$ coordinate frame, $c$ and $z$ represent the translational and rotational parts of the IMU-to-Camera extrinsics expressed in $I$, $\mu_i$ and $\rho_i$ represent the bearing vector and inverse depth parameter to the feature $i$ expressed in the $C$ coordinate frame.

Though the method can track edge features when corner features are unavailable by a modification in the scoring function, the method will fail in a visually featureless
or dark environments. Further details on this method can be found in [24,25].

**Depth-Enhanced ROVIO**

In case the robot has an onboard depth sensor e.g. LiDAR, with a field of view that intersects with that of the camera, LiDAR measurements can be used for the estimation of camera feature depth. The original method of ROVIO is modified to subscribe to pointcloud data and is provided with the LiDAR-to-Camera extrinsics, $T_{CL}$. All points in the pointcloud $P_k$, $p^L_i \in P_k$ are transformed from the LiDAR coordinate frame, $L$, to the Camera coordinate frame, $C$, as

$$p^C_i = T_{CL}p^L_i$$

(2.5)

after which the resulting pointcloud is cropped to the field of view of the camera. The resulting pointcloud is then projected into a depth image, $D_k$ with the same dimensions as the Camera image, using the camera projection model for all points. $D_k$ now corresponds to the same field of view as that of the camera. For each feature $f_i$, a $m \times n$ patch, $m > n$, centered at the feature image coordinates is extracted and the depth, $d$, is calculated as an average of the non-zero values in it. This is used to directly assign a value for the corresponding inverse depth parameter, $\rho_i = 1/d$. In case the direct depth is not available in this manner, the inverse depth is estimated through the procedure in [24].

**ROTIO**

Robust Thermal Inertial Odometry (ROTIO) extends ROVIO by allowing it to work with 14-bit radiometric data. The extension consists primarily of two modifications:

1. Histogram equalization of the incoming thermal image and conversion to 8-bit
for feature detection

2. Modification of the multilevel error minimization for floating point images to allow for operation on the thermal image directly without rescaling

Further details on this method can be found in [26].

2.2 Factor Graphs

Factor graphs are bipartite probabilistic graphical models that represent the relations between two sets of nodes - variables and factors. Each variable node represents a random variable that we would like to estimate while the factor nodes encode the probabilistic information that we have about them. Formally, a factor graph, $F = (U, V, E)$, is composed of the two types of nodes - factors, $\phi_i \in U$, and variables, $x_j \in V$, with the edges, $e_{ij} \in E$, connecting the variables to their associated factors. They define the factorization of a global function $\phi(X)$ as

$$\phi(X) = \prod_i \phi_i(X_i)$$ (2.6)

with the edges essentially encoding the Independence relationships as each factor $\phi_i$ is a function of only the variables $X_i$ adjacent to it. MAP inference on a factor graph can be understood as

$$X^{MAP} = \arg\max_X \phi(X)$$

$$= \arg\max_X \prod_i \phi_i(X_i)$$ (2.7)

Assuming the factors are of the form
\[ \phi_i (X_i) \propto \exp \left\{ -\frac{1}{2} \| h_i (X_i) - z_i \|_{\Sigma_i}^2 \right\} \]  

(2.8)

where \( h_i(X_i) \) denotes the measurement function according to the sensor model and \( z_i \) denotes the sensor measurement, derived from measurements corrupted by zero-mean gaussian noise. Taking the negative log of Equation 2.7 and dropping the \(-1/2\) factor allows us to convert this to a nonlinear least-squares minimization:

\[ X_{MAP} = \arg\min_X \sum_i \| h_i (X_i) - z_i \|_{\Sigma_i}^2 \]  

(2.9)

With a change in the variable, this can then be linearized as

\[ \Delta^* = \arg\min_{\Delta} \sum_i \| A_i \Delta_i - b_i \|_2^2 \]  

(2.10)

with \( \Delta_i = X_i - X_i^0 \), representing the state update vector, \( X_i^0 \) representing the linearization point and \( \Delta^* \) representing the optimized state update vector. By virtue of the design of factor graphs, the matrix \( A \) turns out to be large and sparse allowing the use of fast and efficient methods like CHOLMOD \[27\] and SuiteSparseQR \[28\] tailored to solving sparse systems.

We now briefly describe the different factors used in our approach as well as iSAM2, the method we use for incremental inference.

### 2.2.1 Prior Factor

Prior factors are unary factors that encode our belief about the variables that they are connected to. They can be thought of as anchoring the variable around the value encoded by them. In the case of SLAM, these are generally used in the initialization
of the method to constrain the first state and to provide a prior on landmark locations in case of landmark-based SLAM. The residual error for the combined prior is given by

$$r_p = \begin{pmatrix} \Phi(T_0^{-1}T_p) \\ v_0 - v_p \\ b^a_0 - b^a_p \\ b^\omega_0 - b^\omega_p \end{pmatrix}$$

(2.11)

where $T_0$, $v_0$, $b^a_0$ and $b^\omega_0$ represent the transform, velocity, accelerometer and gyroscope biases of the estimated state, respectively. $T_p$, $v_p$, $b^a_p$ and $b^\omega_p$ represent the prior transform, velocity, accelerometer and gyroscope biases and $\phi : SE(3) \mapsto R^6$ is the lifting operator defined in [29].

### 2.2.2 Combined IMU Factor

For the sake of reducing the computational cost of the optimization by reducing the number of terms involved, IMU preintegration for processing IMU measurements as a single relative constraint was first introduced in [30]. This was developed upon in [17] and [29] where the preintegrated measurement model is given by:

$$\begin{align*}
\Delta \tilde{R}_{ij} &= R^T_i R_j \text{Exp}(\delta \phi_{ij}) \\
\Delta \tilde{v}_{ij} &= R^T_i (v_j - v_i - g \Delta t_{ij}) + \delta v_{ij} \\
\Delta \tilde{p}_{ij} &= R^T_i \left(p_j - p_i - v_i \Delta t_{ij} - \frac{1}{2} g \Delta t_{ij}^2\right) + \delta p_{ij}
\end{align*}$$

(2.12)

for the measurements between times $i$ and $j$ where $\Delta \tilde{R}_{ij}$, $\delta v_{ij}$, $\Delta \tilde{p}_{ij}$ and $\delta \phi_{ij}$, $\Delta \tilde{v}_{ij}$, $\delta p_{ij}$ represent the preintegrated rotation, velocity and position measurements and their corresponding noise.
As the preintegration is carried out in the local frame, the preintegrated measurements are a function of the state that is to be estimated and a random noise. The residual error that is then used in the nonlinear optimization is given by

$$r_{ij} = \begin{bmatrix} r_{\Delta R_{ij}}^\top & r_{\Delta v_{ij}}^\top & r_{\Delta p_{ij}}^\top \end{bmatrix}^\top \in \mathbb{R}^9$$

where

$$r_{\Delta R_{ij}} \doteq \text{Log} \left( \left( \Delta \tilde{R}_{ij}(\tilde{b}_i^q) \right) \text{Exp} \left( \frac{\partial \Delta \tilde{R}_{ij}}{\partial \delta b^q} \delta b^q \right) \right) \left( R_i^T R_j \right)$$

$$r_{\Delta v_{ij}} \doteq R_i^T (v_j - v_i - g \Delta t_{ij})$$

$$- \left[ \Delta \vec{v}_{ij}(\bar{b}_i^q, \bar{b}_i^a) + \frac{\partial \Delta \vec{v}_{ij}}{\partial \delta b^q} \delta b^q + \frac{\partial \Delta \vec{v}_{ij}}{\partial \delta b^a} \delta b^a \right]$$

$$r_{\Delta p_{ij}} \doteq R_i^T (p_j - p_i - v_i \Delta t_{ij} - \frac{1}{2} g \Delta t_{ij}^2)$$

$$- \left[ \Delta \vec{p}_{ij}(\bar{b}_i^q, \bar{b}_i^a) + \frac{\partial \Delta \vec{p}_{ij}}{\partial \delta b^q} \delta b^q + \frac{\partial \Delta \vec{p}_{ij}}{\partial \delta b^a} \delta b^a \right]$$

For the sake of brevity, the derivations for the IMU preintegration are not included here and the reader is referred to [17,29].

### 2.2.3 Pose Between Factor

This factor provides information on and constrains the relative pose between the two associated pose variables subject to the selected noise model. Given the transformation matrices $T_{k-1}^W$ and $T_k^W$ representing the transformations from the world frame to the frame of the particular sensor at time $t_{k-1}$ and $t_k$, their relative transformation is expressed as $T_{rel} = (T_{k-1}^W)^{-1}T_k^W$. As this may not be in the body frame - the coordinate frame used by the factor graph, it needs to be transformed using the body-to-sensor extrinsics, $T_{SB}$ as given by

$$T_{between} = (T_{SB})^{-1}T_{rel}T_{SB} \quad (2.13)$$
The residual can then be defined as

\[ r_s = (T_{\text{between}})^{-1}(X^{-1}_{k-1}X_k) \]  \hspace{1cm} (2.14)

where \( X_{k-1} \) and \( X_k \) represent the estimated state poses at \( t_{k-1} \) and \( t_k \) represented as transformations.

### 2.2.4 iSAM2

iSAM2 \cite{31, 32} is an incremental inference algorithm that leverages the Bayes tree data structure presented in \cite{33}. The method incrementally reorders and relinearizes variables when needed and does not require periodic batch updates. Being an inference method, its primary objective is to estimate a set of variables, \( \Theta \), given a set of nonlinear constraints that are represented as factors, \( \mathcal{F} \) in the factor graph. Furthermore, since in the case of SLAM, new variables, \( \Theta' \), and new factors, \( \mathcal{F}' \) may be added at any point in time, the method should be able to incorporate these into the factor graph and provide an updated solution in real time. Achieving this requires a reduction in the amount of computations being done. iSAM2 achieves this by essentially editing only the parts of the bayes tree structure, \( \mathcal{T} \), that are affected by the new variables and factors since this affected part is typically small. The variable ordering is also done so as to have more recent variables close to the root as they are the most likely to be connected to new measurements as they arrive. The method finally returns \( \Delta \), representing the update to be made to the variables \( \Theta \). While the full details of iSAM2 can be found in \cite{31, 32}, Algorithm 1 lays out the steps that are run for a single iteration of iSAM2.
Algorithm 1 An iteration of iSAM2

1: Add any new factors $\mathcal{F} := \mathcal{F} \cup \mathcal{F}'$
2: Initialize new variables $\Theta'$ and add $\Theta := \Theta \cup \Theta'$
3: Fluid relinearization to yield marked variables $\mathcal{M}$
4: Redo top of bayes tree using $\mathcal{J}$ given as the union of $\mathcal{M}$ and all variables affected by the new factors
5: Solve for $\Delta$
6: New estimate is given by $\Theta \oplus \Delta$
Chapter 3

Proposed Approach

Given the discussion in Chapter 2, we now present our approach on the loose coupling of the two base methods. The architecture of our approach is shown in Figure 3.1.

We define the state of the system in the graph manager as

$$ x_i = [R_i, p_i, v_i, b_i] $$  \hspace{1cm} (3.1)
where $s_i = [R_i, p_i]$ collectively represents the robot pose (orientation and position), $v_i \in \mathbb{R}^3$ represents the robot velocity and $b_i = [b^w_i, b^a_i] \in \mathbb{R}^6$ represents the biases of the gyroscope and accelerometer. Throughout the mission, states are added to the graph and get connected with Pose Between Factors and Combined IMU Factors described in Chapter 2. The resulting nonlinear optimization to be solved is given by

$$X^* = \arg \min_{X} \|r_0\|_0^2 + \sum_i \left(\|r_{I\Delta i}\|_{\Sigma_{I\Delta i}}^2 + \|r_{L\Delta i}\|_{\Sigma_{L\Delta i}}^2 + \|r_{C\Delta i}\|_{\Sigma_{C\Delta i}}^2\right) \quad (3.2)$$

where the terms $r_0$, $r_{I\Delta i}$, $r_{L\Delta i}$, and $r_{C\Delta i}$ represent the residuals of the prior factor, the preintegrated IMU factors and the between factors for the LiDAR scan-to-scan alignment and Visual-/Thermal-Inertial method and the terms $\Sigma_0$, $\Sigma_{I\Delta i}$, $\Sigma_{L\Delta i}$, and $\Sigma_{C\Delta i}$ represent their associated covariances respectively.

Figure 3.2: An instance of the generated factor graph after processing the pointcloud $P_{k=4}$ with the state variables (Purple), the prior factors (Yellow), Combined IMU factors (Red) and two types of between factors - from LiDAR scan-to-scan alignment (Green) and Depth-enhanced ROVIO/ROTIO (Blue)

Figure 3.2 shows an instance of the generated factor graph after the $4^{th}$ pointcloud, $P_{k=4}$, has been processed. The figure illustrates the variables, shown in the circular
nodes, and factors, shown in the square nodes and their respective connections. There are 3 kinds of prior factors represented by $P_s$, $P_v$ and $P_b$, corresponding to the state variables $s_0$, $v_0$ and $b_0$ respectively. Pose between factors from LiDAR scan-to-scan alignment and ROVIO/ROTIO connect sequential poses $s_i$ and $s_{i+1}$ while the Combined IMU factor connect to entire sequential states $x_i$ and $x_{i+1}$.

### 3.1 Algorithm Description

Algorithm 2 presents the pseudocode for the Graph Manager block shown in the system architecture.

---

**Algorithm 2 Graph Management**

1. PropagateStateAndAddIMUFactor($t_{k-1}, t_k$)
2. $T_{k-1,k} \leftarrow \text{GetPriorFromGraph}(t_{k-1}, t_k)$
3. $T_{L,k-1,L_k} \leftarrow T_{LI}T_{L_{k-1,L_k}}T_{IL}$
4. $T_l \leftarrow \text{GetLiDARScanToScanRelativePose}(T_{L,k-1,L_k})$
5. $T_r \leftarrow \text{GetCameraRelativePose}(t_{k-1}, t_k)$
6. if IsRelativeLiDAREstimateGood() then
   7. AddBetweenFactorToGraph($T_l, x_{k-1}, x_k$)
   8. end if
9. if IsRelativeCameraEstimateGood($T_r$) then
   10. AddBetweenFactorToGraph($T_r, x_{k-1}, x_k$)
   11. end if
12. UpdateGraph()
13. $T_o \leftarrow \text{ComputeOptimizedRelativeUpdate}()$
14. PublishOptimizedRelativeTransform($T_o$)

---

The Depth-enhanced ROVIO/ROTIO is provided with the IMU data and Visual/Thermal images to compute and publish a transform on every image while the LiDAR pointclouds are provided to the LOAM odometry step. The LiDAR pointcloud timestamps are also provided to the Graph Manager. A new state, $x_k$, is created in the Graph Manager at the timestamp of each pointcloud, $P_k$. All IMU measurements within $(t_{k-1}, t_k)$ are preintegrated [17] and added as a Combined IMU Factor.
between the states \((x_{k-1}, x_k)\) within the graph after which the graph is updated using iSAM2 \[31\]. The updated graph is then queried for the IMU-to-IMU prior, \(T_{k-1,k}\) which is converted to a LiDAR-to-LiDAR prior using the LiDAR-IMU extrinsics, \(T_{IL}\) and \(T_{LI}\). This transformation prior is used by the LOAM Odometry step to compute the scan-to-scan transformation, which, transformed into the body frame, is stored as \(T_l\). \(T_l\) is optionally added as a between factor in the graph between the states \(x_{k-1}, x_k\) depending on the health metric described in Section 3.1.1. Subsequently, the relative transform \(T_r\) in the body frame between the frames \(VIO_{k-1}^W\) and \(VIO_k^W\) is looked up and optionally added as a between factor into the graph depending on the D-Optimality criterion mentioned in Section 3.1.1.

3.1.1 Health Checking

As shown in Algorithm 2, relative pose measurements from camera and LiDAR sensors need to be independently evaluated before insertion into the combined graph.

LiDAR Health Check

In self similar and geometrically symmetric environments it can be understood that the LiDAR odometry estimation cannot work reliably due to absence of sufficient geometric features. As LiDAR odometry estimates are obtained by minimizing the distance between points during scan–to–scan matching, iterative optimization processes are typically employed to determine the transformation that minimizes the residual error during the two matching steps. Given two corresponding points labelled as measurement \((p_{\text{meas}})\) and prediction \((p_{\text{pred}})\), where \(p_{\text{meas}}\) is the 3D position of the point in the most recent LiDAR scan, based on an initial estimate of the transformation between two scans and given the old position of the corresponding point \((p_{\text{old}})\), \(p_{\text{pred}}\) can be given as:
\[ p_{\text{pred}} = R(p_{\text{old}}) + t, \] (3.3)

where the \( R \) and \( t \) are the initial rotation and translation estimates for the alignment points between pointclouds. These estimates are refined iteratively through an optimization process where the residual distance between \( p_{\text{pred}} \) and \( p_{\text{meas}} \) is minimized, and given as:

\[ r = p_{\text{pred}} - p_{\text{meas}}, \]
\[ e_{\text{res}} = \sum_{i \in P} \|r_i\|_2, \] (3.4)

where \( r \) is the residual between two points and \( e_{\text{res}} \) is the residual cost function to be minimized and contains the squared sum of the residual distances for the set of all corresponding points \( P \) between two pointclouds. Given the relation above, the Gauss-Newton optimization function can be written as:

\[ J^T J \delta x = -J^T r, \] (3.5)

where \( J \) is the Jacobian of the cost function and \( \delta x \) is the transformation increment.

To understand if due to insufficient geometric constraints the underlying scan–to–scan has become degenerate, we evaluate the eigenvalues of the corresponding \( J^T J \) matrices and determine if the underlying optimization process has become degenerate similar to the criteria proposed in [34]. In case the scan–to–scan matching fails and the LiDAR odometry process becomes degenerate, the relative LiDAR pose factor is
Camera Health Check

To check the health of relative visual/thermal-inertial pose estimates, the relative growth of the covariance matrix of VI/TI odometry estimates is measured using the D-Optimality criterion \cite{ref}, which is given as:

\[
D_{Optimality} = \exp(\log(\det(\Sigma^{1/l})))
\]  

(3.6)

where $\Sigma$ is the robot pose covariance matrix with dimensions $l \times l$. 

omitted from graph.
Chapter 4

Evaluation Studies

In this chapter, we describe the conducted evaluation studies and provide an overview of the platforms used for them. The conducted experiments relate to our research preparation for the participation of our team in the DARPA Subterranean Challenge \cite{36} and thus involve relevant environments.

4.1 Platform Description

In the presented experiments, two aerial robotic platforms are utilized to demonstrate the practical utility of the method operating on autonomous platforms in applicable environments. The robots, named “Alpha” and “Charlie”, are both based on the DJI Matrice M100 air frame. Both robots utilize the DJI autopilot to provide low level thrust commands and is connected to an Intel NUC-i7 over a USB-to-serial interface which functions as the high-level processor. The high level processor runs all autonomy functions including sensor data acquisition, the odometry pipeline described in this work, the onboard mapping framework, the exploration path planner, and the platform controller. On both platforms, Model Predictive Control \cite{37} is used to derive control commands which are sent to the M100 autopilot over the USB-to-Serial
interface.

Figure 4.1: The M100 Charlie platform utilized in the presented experiments.

The primary distinction between the two robots is their sensor suite. Both robots feature a FLIR Blackfly BFS-U3-16S2C color camera, obtaining images in $720 \times 540$ resolution at $20Hz$, and a VectorNav VN100 Inertial Measurement Unit (IMU) providing measurements at $200Hz$. Both platforms are also equipped with Cree XHP70 LEDs which are synchronized with the camera’s shutter to provide illumination in low light environments. Alpha is equipped with a Velodyne Puck Lite VLP-16 LiDAR providing 30,000 range measurements at $10Hz$ within a 30 deg vertical Field of View (FoV). Charlie is equipped with an Ouster OS-1 LiDAR, providing 131,000 range measurements at $10Hz$ within a 45 deg vertical FoV, and a FLIR Tau2 Thermal Camera which provides $640 \times 512$ resolution at $30Hz$.

4.2 Experimental Results

We conducted two field experiments that are presented here, namely inside a man-made urban tunnel and in an underground gold mine.
4.2.1 Tunnel Test

We first test our method in a highway underpass in Northern Nevada, USA using the Alpha robot. Due to the self similarity and lack of geometric features in this environment, the scan-to-scan alignment of the LiDAR only method (LOAM) is unable to fully constrain the solution and as can be seen in Figure 4.2, LOAM fails to accurately capture the length of the tunnel which is reported by it as 26.68m. In the proposed
method, as the LOAM Odometry result is selectively integrated and since the VIO solution is healthy due to sufficient visual features, the optimized result provided to the LOAM Mapping step results in a significant improvement in building the map of this environment with the tunnel length being reported as 70.08m.

4.2.2 Underground Mine Test

For the field-verification of our method, the Charlie robot is deployed in the Turquoise-Ridge Joint Venture (TRJV) mine in Winnemucca, Nevada, USA. A previous work form our lab, GBPlanner [1, 38], is used for the autonomous planning of the robot which allows it to explore the mine maximizing for volumetric exploration. After take-off, the robot navigates by locally maximizing for the exploration of unknown space and thus travels through multiple intersections, which are saved as frontiers for future repositioning to continue exploration when a dead end is met. Once the remaining charge in the battery of the robot has reduced to below a specified threshold (visualized in Figure 4.3 as point (c)), GBPlanner provides a return-to-home path bringing the robot to the initial take-off location and completing a trajectory that is approximately 410m in length. As the ground-truth is not available for this experiment, the return of the robot to the take-off location is indicative of minimal drift in the proposed solution.
Figure 4.3: The top row of this figure shows the LiDAR-Thermal-Inertial odometry and mapping result in a deployment of the “Charlie” robot in an active underground gold mine in Northern Nevada, USA. Charlie is tasked to explore the environment using [1] and is later commanded to return-to-home. The accuracy of the approach is verified by virtue of the fact that the robot returns to the take-off spot to land. The middle and bottom rows illustrate the tracked Thermal-Inertial features and the environmental conditions at the specific labelled instances in the mission.
Chapter 5

Conclusions and Future Work

This thesis presented a method for a loosely coupled LiDAR-Visual/Thermal-Inertial Odometry and Mapping system that leverages factor graphs. The method performed the fusion of relative pose measurements from a visual/thermal method and a LiDAR scan-to-scan alignment method as well as preintegrated IMU measurements within a factor graph that was updated incrementally using iSAM2 \[31\]. The optimized relative pose is then fed into a fine LiDAR scan-to-map alignment method that finally provides the optimized odometry of the system along with a registered pointcloud. The method is tested in a self-similar environment and field-verified in an underground mine deployment.

5.1 Future Work

Due to the easily-extendible nature of factor graphs, we intend to continue development and extend this work in several directions. Leveraging the inherent benefits of joint optimization, this method could be enhanced by integrating different sensor-specific front-ends directly with the graph manager as a backend. For example, given our lab’s recent work on Keyframe-based Thermal Inertial Odometry \[20\], its front-
end coupled with a landmark management module can be used to add landmarks directly into the optimization along with LiDAR constraints for a tight Thermal-Inertial-LiDAR approach. Constraints like this from multiple different sensors could be added into the optimization, subject to the physical and computational limitations of the robot.

The DARPA Subterranean Challenge requires robots to be sent into unknown environments to reconstruct them accurately as well as locate certain objects of interest, termed as artifacts, within a certain degree of accuracy. These artifacts may require different modalities to be detected. For instance, a visual camera could be used to detect a backpack or fire extinguisher, a Bluetooth or WiFi module could be used to detect a phone and a thermal camera can be used to detect a hot Vent. As a single detection may not provide an accurate estimate of the location of the artifact, an additional variable could be added into the graph for the location of the artifact which is updated by factors generated through multiple detections.
References


