Stereo Visual-Inertial-LiDAR Simultaneous Localization and Mapping

Weizhao Shao

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

June 2019

CMU-RI-TR-19-48

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Simultaneous Localization and Mapping (SLAM) is a fundamental task to mobile and aerial robotics. The goal of SLAM is to utilize on-board sensors for estimating the robot's trajectory while reconstructing the surrounding environment (map) in real-time. The algorithm should also be able to perform loop closure, such that it could detect if same environments are revisited again and hence eliminate drifts over the loop. SLAM has been an appealing field of research over the past decades, for one it is a great mix of probabilistic estimation, optimization, and geometry. For two, it is practically useful but hard, as it involves tasks from sensor calibrations to system integration.

The community has been investigating different sensor modalities and exploiting their benefits. LiDAR-based systems have proven to be accurate and robust in most scenarios. However, pure LiDAR-based systems fail in certain degenerate cases like traveling through featureless tunnels or straight hallways. Vision-based systems are efficient and lightweight. However, they depend on good data associations to perform well, and thus fail terribly in environments without many visual clues. Inertial Measurement Unit (IMU) produces high-frequency measurements, which are reasonable for a short interval but quickly drift.

In this thesis, I investigate the fusion of LiDAR, camera and IMU for SLAM. I will begin with my implementation of a stereo visual inertial odometry (VIO). Then I will discuss two coupling strategies between the VIO and a LiDAR mapping method. I will also present a LiDAR enhanced visual loop closure system to fully exploit the benefits of the sensor suite. The complete SLAM pipeline generates loop-closure corrected 6-DOF LiDAR poses in real-time and 1cm voxel dense maps near real-time. It demonstrates improved accuracy and robustness compared to state-of-the-art LiDAR methods. Evaluations are performed on representative public datasets and custom collected datasets from diverse environments.

Acknowledgements

I want to express my sincere gratitude to my advisor, Professor George Kantor, for his belief in my potential, unconditional supports, and valuable discussions along my great two years at Carnegie Mellon University. His profound engineering and scientific insights have enlightened me so many times and will continue to be influential in my life. I would not be in the position where I am right now if it were not for him.

I want to say a great thank you to my lab mates, Cong Li and Srinivasan Vijayarangan. Having the opportunities to work closely with them is one of the greatest opportunities that I could have ever asked for. they demonstrate me professionalism, productivity, and critical thinking, which I will carry along with in my career.

Finally, I want to sincerely thank my families and friends. You have given me the greatest understanding and support along the two years. You have always encouraged me to pursue my academic and life goals. You have shared my joy and sorrow, love and pain. Having you in my life makes me feel favoured by fortune.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

Enabling mobile robots to perform tasks and interact with people in real world has been a long-term goal that drives scientists and engineers along their way. One of the fundamental tasks to solve is to give a robot the sense of position and environments. SLAM algorithms are to tackle this fundamental task. The ability to solve a SLAM problem is essential to practice. High-level robotics tasks relies on it: A planning system needs to know the robot's pose to do planning and avoid obstacles. A semantic scene segmentation system needs to have some map representation as its input data. More generally, applications like search and rescue, AR (augmented reality), autonomous driving, and indoor mobile robots all requires doing SLAM in one form or another. SLAM problems are also interesting theoretical. Evolving for several decades, this domain has become a great mix of probabilistic estimation, geometry, signal processing, and optimization. With cameras becoming popular and more visual SLAM algorithms being developed, the community also establishes a deep connection with the field of computer vision and methods such as Bundle Adjustment (BA). Though most SLAM problems are considered solved, current SLAM algorithms still lack the robustness to be freely deployed in real world, as argued by Cadena et al. [7]. The abilities to recover from failure, and deal with challenges from environment, non-trivial robot's motion, and sensor failures still require a vast amount of algorithmic efforts. Not to mention that practical issues such as synchronization and sensor calibrations must be treated well.

A multi-sense fusion approach naturally comes in mind when thinking about improving the robustness. Different sensors have their own superiority and failure mode. There have been lots of efforts on exploring different sensor modalities. LiDAR has been a popular sensor in the robotics community. LiDAR based SLAM systems have demonstrated their advantages in being accurate and robust in most cases. However, pure LiDAR approaches fail in certain degenerate cases like traveling through featureless tunnels or straight hallways. Vision based systems are efficient and lightweight. However, the range of observations is limited and they depends on good data associations to perform well. Thus they fail terribly in environments without much visual clues. Inertial Measurement Unit (IMU) produces high frequency measurements, which are reasonable for a short interval with biases being corrected, but drift quickly. With an integration of the three where their advantages could still be fully exploited, they compliment each other to deal with individual sensor failure. The formulation of the estimation problem also favors this approach. Sensor inputs are modeled as independent and identically distributed observations in the estimator, and adding a sensor is hence as straightforward as adding an observation once the sensor model and the noise model are available.

This thesis aims to explore this multi-sense approach for improving the robustness of a SLAM pipeline. Specifically, I investigate the fusion of LiDAR, camera and IMU for tackling the real-time state estimation and mapping problem. I will begin with the factor graph formulation and implementation of a feature-based stereo visual inertial odometry (VIO). Then I will discuss two coupling strategies between the VIO and a state-of-the-art LiDAR mapping method. I will also present a LiDAR enhanced visual loop closure system, which consists of a global factor graph optimization, to fully exploit the benefits of the sensor suite. The complete SLAM pipeline, Visual-Inertial-LiDAR SLAM (VIL-SLAM), is able to generate loop-closure corrected 6-DOF poses in real-time and dense 1cm voxel-size maps near real-time. It demonstrates improved accuracy and robustness in challenging environments, where state-of-the-art LiDAR methods fail easily. Evaluations are performed on representative public datasets and custom collected datasets from diverse environments.

1.2 Related Work

Visual Inertial Odometry An odometry differs from a SLAM pipeline as it does not have a loop closure system to correct global drift. The output is smooth, and is preferred for applications who only consider local motions. Because no global information is required to retain for global drift correction, the memory consumption and time complexity generally remain the same as the operation time grows long. Current VIO literature introduces various formulations to integrate visual and inertial data. The literature characterizes different approaches into *tightly-coupled system* [40, 29, 23], in which visual information and inertial measurements are jointly optimized, or *loosely-coupled system* [16, 32, 51], in which IMU is a separate module and fused with a vision-only state estimator. The approaches could be further divided into either filtering-based [47, 41, 3, 52, 51, 17] or graph-optimization based [40, 29, 23, 24, 48]. Filtering-based approaches use nonlinear filtering techniques, and generally suffer from poor linearization point. Tightly-coupled optimization-based approaches, using an Maximum a posteriori (MAP) formulation and taking the benefit of minimizing residuals iteratively, usually achieve better accuracy and robustness with a higher computation cost. However, we see that some VIO systems based on filtering have also demonstrated state-of-the-art performance. Some representative examples include the VIN systems of Kottas et al. [27], Hesch et al. [22], and the MultiState Constraint Kalman Filter of Mourikis and Roumeliotis [35]. They reduce the performance gap between filtering and MAP estimation by taking care of potential sources of inconsistency and getting more accurate linearization point. In this work, I have implemented a stereo VIO systems using a fixed-lag smoothing framework as the backend. It follows a tightly-coupled formulation and uses marginalization to bound the computation costs to achieve the real-time performance.

Visual SLAM Visual SLAM algorithms aim to recover a globally consistent representation for both robot's pose and the map. Compared with LiDAR-based SLAM algorithms, they become popular for the accessibility from small mobile devices, which typically have limited computation resources. PTAM [26] is considered as the earliest modern visual SLAM system. It proposes the idea of tracking a camera and mapping the environment in parallel, which makes these algorithms accurate and efficient enough for real-time applications. The tracking part is responsible for local motion recovery and providing a real-time estimate for the camera's pose. The mapping part runs a global BA for achieving the global consistency in the back. This idea is widely adopted in future works, such as DTAM [37], ORB-SLAM [42], and SOFT-SLAM [11]. In VIL-SLAM, I adapt this idea to using VIO as the tracking thread and using a LiDAR mapping method for maintaining global consistency. In this way, advantages of both methods are exploited: VIO is able to provide accurate local motion estimate and LiDAR scan to map matching could reliably register scans to global map, at the same time correcting local drift. From another perspective on a visual SLAM system, the ability to do loop closure (loop detection and loop validation) directly distinguish it with a VIO. The bag-of-words method [20] is a widely adopted feature-based approach for loop detection. Though it suffers from severe illumination variations, it demonstrates reliable performance when the training set is carefully constructed. Several other methods are developed to boost the performance via matching sequences [33], incorporating spatial and appearance information [28], and unifying different visual appearances into one representation [10]. Lowry et al. [30] provides a comprehensive survey on visual place recognition methods. Loop validation is for preventing wrong loop detection results to get into the estimator. A common method in vision-based approaches is to use RANSAC for outlier rejection [43]. In VIL-SLAM, I adopt the bag-of-words approach for visual loop detection and a geometric verification method based on RANSAC for loop validation. Furthermore, I introduce LiDAR into the loop closure system for further loop validation and loop constraint refinement.

LiDAR-based SLAM Current state-of-the-art SLAM systems using just laser scanner are [9, 49, 50, 53, 14], in which a motion model is required, either a constant velocity model or a Gaussian process. These approaches are variants of iterative closest point (ICP) and rely on robust data association to perform well. LOAM [53], the top ranked LiDAR-only method on KITTI odometry benchmark [21], introduces the separation of a LiDAR odometry and a LiDAR mapping to tackle the problem. The former conducts a scan to scan matching to estimate local motion while the latter runs a scan to map registration using the estimated local motion as the prior. The goal of the local motion estimate is to provide a relatively good initialization point, which is essential for the scan to map ICP optimization to perform well and converge faster. In VIL-SLAM, I couple my VIO implementation with the LiDAR mapping portion of LOAM to correct the local drift. A VIO provides a better initialization point for the ICP optimization: Experiments show that the motion estimate from a VIO is much accurate compared to a LiDAR odometry. Also, a VIO uses IMU as motion model

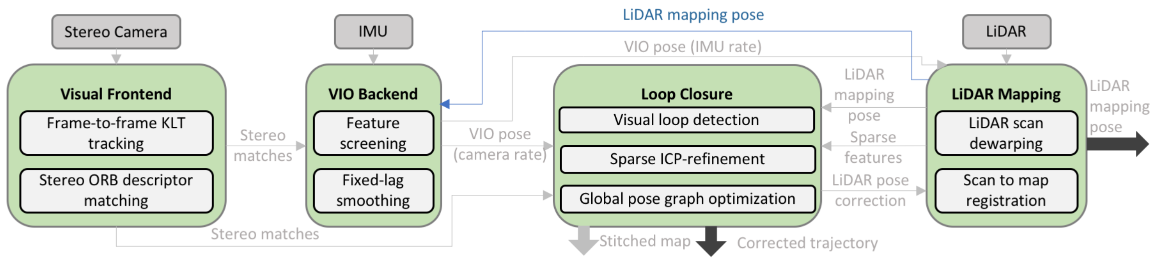


Figure 1.1: The system diagram of VIL-SLAM. Sensors are in gray and system modules are in green. Arrows indicate how messages flow within the system. Blue arrow indicates that this message is optional, depending on the system configuration. The dark thick arrows indicate the system real-time output and the light thick arrow indicates the output generated in post-processing near real-time.

and could provide estimate up to IMU rate, eliminating the need for a constant velocity motion model in LiDAR odometry, which is less accurate in comparison. Approach in [15] combines stereo cameras and a laser scanner. It has motion estimate generated from a visual odometry (VO) and refined by matching laser scans. The differences to VIL-SLAM are that they use multi-resolution grid map representation, but ours uses sparse point cloud for localization and outputs dense point cloud as map representation. Also, VIO is usually more robust and accurate compared to a VO [13].

VLOAM [54], which uses an IMU, a monocular camera, and a laser scanner is the most similar existing system to VIL-SLAM. However, they are different from several perspectives. VLOAM [54] loosely couples IMU, camera, and LiDAR. It uses a sequential pipeline to process the sensor data: High-rate IMU data provides motion model for the visual odometry, which then initializes the LiDAR ICP registration. Both camera and LiDAR are able to provide a feedback for correcting the IMU biases. LiDAR points are projected into the image frame, serving as depth information. This pipeline is robust to either camera or LiDAR failure, as the other one could still correct the IMU biases and generate motion estimate. In comparison, VIL-SLAM uses a tightly-coupled VIO as the motion model to initialize the LiDAR mapping algorithm instead of a loosely-coupled one. For LiDAR, both loosely-coupling and tightly-coupling strategies are proposed, and the latter feeds the LiDAR mapping results back into the VIO smoother as an additional constraint, making the smoother a local batch optimization consisting of information from all three sensors. This guarantees an accurate local motion estimation which is also robust to either camera or LiDAR failure. Moreover, all sensory information are coupled together, which makes the optimization problem better constraint. In terms of correcting drift after long traversal, VIL-SLAM has a LiDAR enhanced loop closure system whereas VLOAM only depends on the LiDAR mapping system to match scans to existing map when there is a loop.

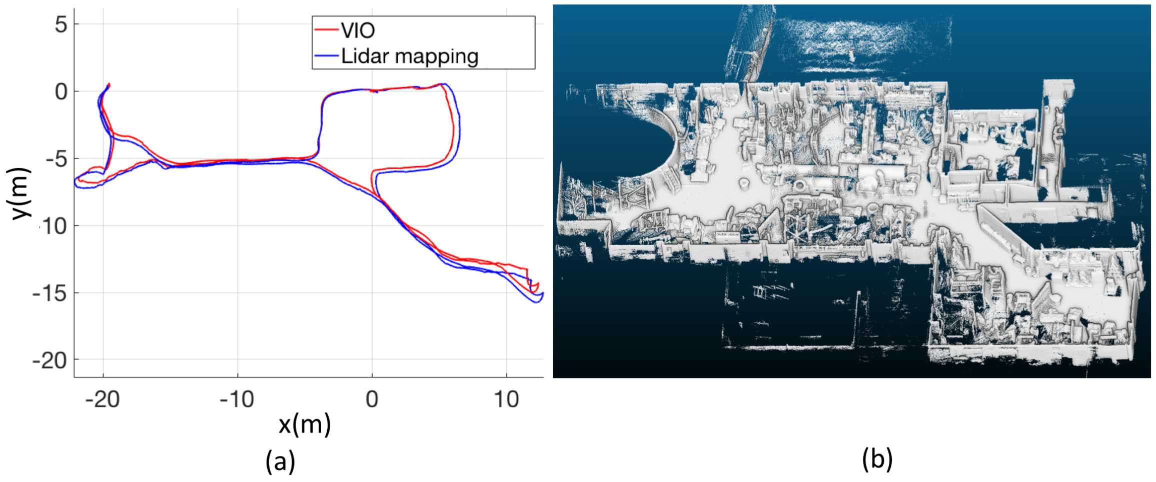


Figure 1.2: Sample outputs of VIL-SLAM from an indoor test. LiDAR feedback is closed and loop closure system is turned off in this test. (a) Pose estimate results from both the VIO and LiDAR mapping systems obtained in real-time. (b) Dense mapping results (cross-sectioned) stitched with LiDAR mapping pose estimate.

1.3 System Overview

VIL-SLAM consists of three major systems, which are a stereo VIO, a LiDAR mapping system, and a LiDAR enhanced visual loop closure system. They collectively exploit the advantages of IMU, stereo camera, and LiDAR. Fig. 1.1 shows the complete pipeline of VIL-SLAM. The stereo VIO consists of a visual frontend and a backend optimizer. The visual frontend takes stereo pairs from the stereo cameras. It performs frame to frame tracking and stereo matching, and outputs stereo matches as visual measurements. The backend optimizer takes stereo matches and IMU measurements, performs IMU pre-integration and tightly-coupled fixed-lag smoothing over a pose graph. When VIL-SLAM is configured to perfrom LiDAR feedback, the pose graph has one additional constraint added, which is the pose between factor formulated from the LiDAR mapping poses. The optional blue arrow in Fig. 1.1 corresponds to this configuration. The VIO backend optimizer outputs pose estimate at both IMU rate and camera rate in real-time. The LiDAR mapping system follows the one in LOAM [53]. It uses the motion estimate from the VIO and performs LIDAR points dewarping and scan to map registration. The LiDAR enhanced visual loop closure system conducts visual loop detection and initial loop constraint estimation, which is further validated by RANSAC geometric verification and refined by a sparse point cloud ICP alignment. A global pose graph constraining all LIDAR poses is optimized incrementally to obtain a globally corrected trajectory and a LIDAR pose correction in real-time once there is a loop. They are sent back to LIDAR mapping module for map update and re-localization. In post processing, the logged dewarped LIDAR scans are stitched with the best LIDAR pose estimate to have the dense mapping results as in Fig. 1.2. The method assumes extrinsic and intrinsic calibration parameters are obtained beforehand.

2 Stereo Visual Inertial Odometry

2.1 Hybrid Visual Frontend

Visual frontend accepts a stereo pair, and performs frame to frame feature tracking and stereo matching for the generation of a set of stereo-matched sparse feature points, namely, stereo matches. A stereo match could either be one tracked from previous stereo pair, or a new one extracted in this pair. The frame to frame tracking performance directly affects the quality of temporal constraints. Stereo matching is important for bootstrapping the system and generating high-quality matches to constrain the scale. These two tasks are crucial for any stereo visual frontend. Traditionally in a feature-based visual frontend, both of the tasks are performed in descriptor space. However, we note that this method is sensitive to parameter tuning and time-consuming. More importantly, this method does not use the prior information (previous frame) in the tracking task. Direct methods show robust and efficient temporal tracking results in recent years [16, 18]. Hence, I use Kanade Lucas Tomasi (KLT) feature tracker [4] to track all feature points of the previous stereo matches, either in the left or right image. Only when they are both tracked, we have a tracked stereo match and it is pushed into the output. For stereo matching task, I still use feature-based methods which are better suited to handle large baselines than KLT. Large stereo baseline helps scale estimation and reduces degeneracy issues caused by distant features. Since the system combines direct and descriptor space methods, hence become a hybrid of the two. If the number of tracked stereo matches is below a threshold, we perform feature extraction using Shi-Tomashi Corner detector [44], followed by a feature elimination process in which features that have pixel coordinate distance to any existing features smaller than a threshold are deleted. ORB (Oriented FAST and Rotated BRIEF) [42] descriptors are then computed on all survived features, followed by a brute-force stereo matching to obtain new stereo matches. The system initializes by performing stereo matching on the first stereo pair.

2.2 Backend Optimizer

The goal of the backend optimizer is to provide real-time locally consistent state estimate at a relatively high frequency, serving as the motion model for the LIDAR mapping algorithm. A tightly-coupled fixed-lag smoother operating over a pose graph is a good trade-off between accuracy and efficiency. Optimization-based methods in general allow for multiple re-linearization to approach the global minimum. A fixed-lag pose graph optimizer further bounds the maximum number of variables, and hence the computation cost is bounded. Since bad visual measurements cause convergence issues, we enforce a strict outlier rejection mechanism on visual measurements. The system eliminates outliers by checking the average reprojection error,

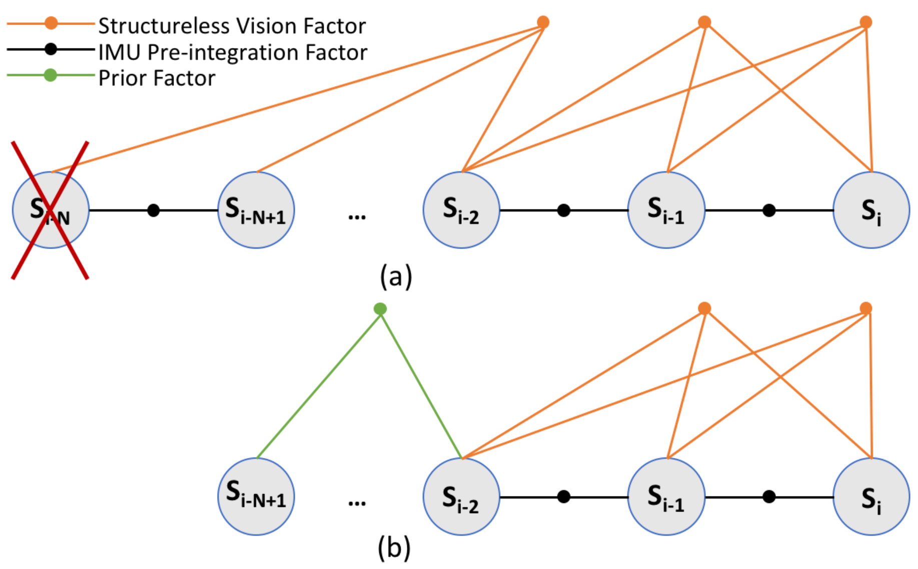


Figure 2.1: Fixed-lag pose graph formulation in the VIO. State variables being optimized are circled, where i stands for the current state and N is the window size. (a) The state to be marginalized is crossed. (b) After marginalization, prior factors are added back on related variables.

both stereo and temporal. Another advantage of formulating the problem as a pose graph optimization problem is that it unifies different kinds of observations into the factor representation. This eases the the procedure of adding new sensor inputs or constraints into the optimization problem. The coupling of LiDAR information introduced in the next section demonstrates this perspective.

The VIO proposed has *IMU Pre-integration Factor* and *Structureless Vision Factor* as constraints in default, and has *Pose Between Factor* when LiDAR feedback is configured. The pose graph formulation without LiDAR feedback is shown in Fig. 2.1. Variables to be optimized are the states inside the window. Denote $\mathbf{S}_t$ as the state variable at the stereo frame time t, we have

$$\mathbf{S}_t \doteq [\xi_t, \mathbf{v}_t, \mathbf{b}_t^a, \mathbf{b}_t^g]$$

where $\xi_t \doteq (\mathbf{R}_t, \mathbf{p}_t) \in SE(3)$ represents the 6 Degrees of Freedom (DoF) system pose (IMU-centered robot pose at time t). $\mathbf{v}_t \in \mathbb{R}^3$ is the associated linear velocity. $\mathbf{b}_t^a \in \mathbb{R}^3$ and $\mathbf{b}_t^g \in \mathbb{R}^3$ are the accelerometer bias and gyroscope bias respectively. The state variables being estimated at time t are those inside the sliding window consisting of the most recent N keyframes

$$\{\mathbf{S}_t\}_N$$

Past state variables are marginalized by the estimator. This procedure produces prior factors on related variables which are still inside the window.

2.2.1 IMU pre-integration factor

I follow the IMU pre-integration theory introduced in [19, 8] to process IMU measurements. The idea of pre-integrating IMU measurements is first introduced in [31] to ease the computational cost. It adopts Euler angles as global parametrization for rotations and hence has the issue of singularities and inconsistency of rigid transformation in Euclidean space [34]. The pre-integration theory is built up on this work and addresses the manifold structure of the rotation group. It preserves the original insight of pre-integration, that is, the integration is performed locally and does not require repetitive integration when the linearization point in the optimization changes. Moreover, the theory provides a complete theoretical derivation for all Jacobians, uncertainty propagation, and a-posteriori bias correction. I present here only the key steps in writing the pre-integration factor. Readers should refer to [19] for the details.

For two consecutive states $\mathbf{S}_i$ and $\mathbf{S}_j$, denote $k = i, \cdots, j$ as the IMU measurements in between, the relative motion increments that are independent of the pose and velocity could be derived as

$$\begin{aligned}
\Delta\mathbf{R}_{ij} &\doteq \mathbf{R}_i^{\intercal}\mathbf{R}_j = \prod_{k=i}^{j-1} \exp\left((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g - \boldsymbol{\eta}_k^{gd})\Delta t\right) \\
\Delta\mathbf{v}_{ij} &\doteq \mathbf{R}_i^{\intercal}(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) = \sum_{k=i}^{j-1} \Delta\mathbf{R}_{ik}(\tilde{\boldsymbol{a}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^{ad})\Delta t \\
\Delta\mathbf{p}_{ij} &\doteq \mathbf{R}_i^{\intercal}(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2) \\
&= \sum_{k=i}^{j-1} \left[\Delta\mathbf{v}_{ik}\Delta t + \frac{1}{2}\Delta\mathbf{R}_{ik}(\tilde{\boldsymbol{a}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^{ad})\Delta t^2\right] \\
&= \sum_{k=i}^{j-1} \left[\frac{3}{2}\Delta\mathbf{R}_{ik}(\tilde{\boldsymbol{a}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^{ad})\Delta t^2\right]
\end{aligned} \tag{2.1}$$

In which $\tilde{\boldsymbol{\omega}}_k$ and $\tilde{\boldsymbol{a}}_k$ are the measured angular velocity from gyroscope and acceleration from accelerometer, $\boldsymbol{\eta}_k^{gd}$ and $\boldsymbol{\eta}_k^{ad}$ are the modeled discrete-time white noise for IMU (additive for gyroscope and accelerometer individually), $\Delta t_{ij} = sum_{k=i}^{j}\Delta t_k$, $\Delta\mathbf{R}_{ik} \doteq \mathbf{R}_i^{\intercal}\mathbf{R}_k$, $\Delta\mathbf{v}_{ik} = \mathbf{v}_k - \mathbf{v}_i$. Note that the biases between two consecutive keyframes $\mathbf{S}_i$ and $\mathbf{S}_j$ are modeled as constant. However, the equations still depend on the biases. So when the biases change during optimization, potential re-integration is required. To proceed, the biases are first assumed given, and then the authors show how to avoid the re-integration.

Using first-order approximations of the exponential map and the *Adjoint* representation of the exponential map

$$\exp(\phi^{\wedge}) \approx I + \phi^{\wedge}$$

$$\mathrm{Exp}(\phi + \delta\phi) \approx \mathrm{Exp}(\phi)\mathrm{Exp} J_r(\phi)\delta\phi$$
$$\mathrm{Exp}(\phi)\mathrm{R} = \mathrm{R}\mathrm{Exp}(\mathrm{R}^\intercal)$$

The approximated measurements are derived

$$\begin{aligned}
\Delta\mathbf{R}_{ij} &\doteq \Delta\tilde{\mathbf{R}}_{ij}\mathrm{Exp}(-\delta\phi_{ij}) \\
\Delta\mathbf{v}_{ij} &\doteq \Delta\tilde{\mathbf{v}}_{ij} - \delta\mathbf{v}_{ij} \\
\Delta\mathbf{p}_{ij} &\doteq \Delta\tilde{\mathbf{p}}_{ij} - \delta\mathbf{p}_{ij}
\end{aligned} \tag{2.2}$$

in which

$$\Delta\tilde{\mathbf{R}}_{ij} = \prod_{k=1}^{j-1} \mathrm{Exp}\big((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g)\Delta t\big)$$

$$\mathrm{Exp}(-\delta\phi_{ij}) = \prod_{k=1}^{j-1} \mathrm{Exp}\big(\Delta\tilde{\mathbf{R}}_{(k+1)j}^\intercal J_r^k \boldsymbol{\eta}_k^{gd}\Delta t\big)$$

$$\Delta\tilde{\mathbf{v}}_{ij} = \sum_{k=i}^{j-1} \Delta\tilde{\mathbf{R}}_{ik}(\tilde{\boldsymbol{a}}_k - \mathbf{b}_i^a)\Delta t$$

$$\delta\mathbf{v}_{ij} = \sum_{k=i}^{j-1} \big[\Delta\tilde{\mathbf{R}}_{ik}\boldsymbol{\eta}_k^{ad}\Delta t - \Delta\tilde{\mathbf{R}}_{ik}(\tilde{\boldsymbol{a}}_k - \mathbf{b}_i^a)^\wedge \delta\phi_{ik}\Delta t\big]$$

$$\Delta\tilde{\mathbf{p}}_{ij} = \sum_{k=i}^{j-1} \frac{3}{2}\Delta\tilde{\mathbf{R}}_{ik}(\tilde{\boldsymbol{a}}_k - \mathbf{b}_i^a)\Delta t^2$$

$$\delta\mathbf{p}_{ij} = \sum_{k=i}^{j-1} \big[\frac{3}{2}\Delta\tilde{\mathbf{R}}_{ik}\boldsymbol{\eta}_k^{ad}\Delta t^2 - \frac{3}{2}\Delta\tilde{\mathbf{R}}_{ik}(\tilde{\boldsymbol{a}}_k - \mathbf{b}_i^a)^\wedge \delta\phi_{ik}\Delta t^2\big]$$

$\Delta\tilde{\mathbf{R}}_{ij}$, $\Delta\tilde{\mathbf{v}}_{ij}$, and $\Delta\tilde{\mathbf{p}}_{ij}$ are the pre-integrated rotation, velocity, and position measurements. $\delta\phi_{ij}$, $\delta\mathbf{v}_{ij}$, and $\delta\mathbf{p}_{ij}$ are the associated noises.

To deal with the bias update $b \leftarrow \bar{b} + \delta b$ during the optimization, first-order approximation is used to update the delta measurements:

$$\begin{aligned}
\Delta\tilde{\mathbf{R}}_{ij}(\mathbf{b}_i^g) &\approx \Delta\tilde{\mathbf{R}}_{ij}(\bar{\mathbf{b}}_i^g)\mathrm{Exp}\left(\frac{\partial\Delta\bar{\mathbf{R}}_{ij}}{\partial\mathbf{b}_i^g}\delta\mathbf{b}_i^g\right) \\
\Delta\tilde{\mathbf{v}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a) &\approx \Delta\tilde{\mathbf{v}}_{ij}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \frac{\partial\Delta\bar{\mathbf{v}}_{ij}}{\partial\mathbf{b}_i^a}\delta\mathbf{b}_i^a + \frac{\partial\Delta\bar{\mathbf{v}}_{ij}}{\partial\mathbf{b}_i^g}\delta\mathbf{b}_i^g \\
\Delta\tilde{\mathbf{p}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a) &\approx \Delta\tilde{\mathbf{p}}_{ij}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \frac{\partial\Delta\bar{\mathbf{p}}_{ij}}{\partial\mathbf{b}_i^a}\delta\mathbf{b}_i^a + \frac{\partial\Delta\bar{\mathbf{p}}_{ij}}{\partial\mathbf{b}_i^g}\delta\mathbf{b}_i^g
\end{aligned} \tag{2.3}$$

I omit the Jacobian's expression which could be found in the original paper. Note that the Jacobian is constant across the optimization and can be computed beforehand.

The biases are modeled to be slowly-varying in the model, hence with a "Brownian motion", that is, an integrated white noise:

$$\dot{\mathbf{b}}^g(t) = \boldsymbol{\eta}^{bg} \qquad \dot{\mathbf{b}}^a(t) = \boldsymbol{\eta}^{ba}$$

By integration, for two consecutive states $\mathbf{S}_i$ and $\mathbf{S}_j$, the following could be written:

$$\begin{aligned} \mathbf{b}_j^g &= \mathbf{b}_i^g + \boldsymbol{\eta}^{bgd} \\ \mathbf{b}_j^a &= \mathbf{b}_i^a + \boldsymbol{\eta}^{bad} \end{aligned} \tag{2.4}$$

where $\boldsymbol{\eta}^{bgd}$ and $\boldsymbol{\eta}^{bad}$ are the discrete noises which have zero mean and covariance $\boldsymbol{\Sigma}^{bgd} \doteq \Delta t_{ij}\mathrm{Cov}(\boldsymbol{\eta}^{bg})$ and $\boldsymbol{\Sigma}^{bad} \doteq \Delta t_{ij}\mathrm{Cov}(\boldsymbol{\eta}^{ba})$ [19]. This equation is readily to be used for formalizing a residual term for the bias update. Together with the previous derivation, we have the residuals represented by the IMU pre-integration factor, $\mathbf{r}_{I_{ij}} = [\mathbf{r}_{\Delta\mathbf{R}_{ij}}^\intercal, \mathbf{r}_{\Delta\mathbf{v}_{ij}}^\intercal, \mathbf{r}_{\Delta\mathbf{p}_{ij}}^\intercal, \mathbf{r}_{\mathbf{b}_{ij}}^\intercal]^\intercal$, derived:

$$\begin{aligned} \mathbf{r}_{\Delta\mathbf{R}_{ij}} &\doteq \mathrm{Log}\Bigg(\big(\Delta\tilde{\mathbf{R}}_{ij}(\bar{\mathbf{b}}_i^g)\mathrm{Exp}\big(\frac{\partial\Delta\bar{\mathbf{R}}_{ij}}{\partial\mathbf{b}_i^g}\delta\mathbf{b}_i^g\big)\big)^\intercal\mathbf{R}_i^\intercal\mathbf{R}_j\Bigg) \\ \mathbf{r}_{\Delta\mathbf{v}_{ij}} &\doteq \mathbf{R}_i^\intercal(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) \\ &\quad - \left[\Delta\tilde{\mathbf{v}}_{ij}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \frac{\partial\Delta\bar{\mathbf{v}}_{ij}}{\partial\mathbf{b}_i^a}\delta\mathbf{b}_i^a + \frac{\partial\Delta\bar{\mathbf{v}}_{ij}}{\partial\mathbf{b}_i^g}\delta\mathbf{b}_i^g\right] \\ \mathbf{r}_{\Delta\mathbf{p}_{ij}} &\doteq \mathbf{R}_i^\intercal(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2) \\ &\quad - \left[\Delta\tilde{\mathbf{p}}_{ij}(\bar{\mathbf{b}}_i^g, \bar{\mathbf{b}}_i^a) + \frac{\partial\Delta\bar{\mathbf{p}}_{ij}}{\partial\mathbf{b}_i^a}\delta\mathbf{b}_i^a + \frac{\partial\Delta\bar{\mathbf{p}}_{ij}}{\partial\mathbf{b}_i^g}\delta\mathbf{b}_i^g\right] \\ ||\mathbf{r}_{\mathbf{b}_{ij}}||^2 &= ||\mathbf{b}_j^g - \mathbf{b}_i^g||_{\boldsymbol{\Sigma}^{bgd}}^2 + ||\mathbf{b}_j^a - \mathbf{b}_i^a||_{\boldsymbol{\Sigma}^{bad}}^2 \end{aligned} \tag{2.5}$$

2.2.2 Structureless vision factor

In a traditional factor graph based VIO, visual landmarks are modeled as variables in the graph, either in inverse depth or 3D position representation. Visual measurements are then the camera projection factors. In this work, visual landmarks are not explicitly modelled, and visual measurements are modeled in a structureless fashion, similar to [19, 36, 5]. The benefits of the latter approach are two-fold. First, variable size is bounded to be the sliding window size at any point in time. The computational cost is hence bounded. Second, it is easier to manage the landmark variables. When a keyframe is out of the window, landmarks of all observations associated with it are marginalized automatically.

Consider a landmark p, whose position in global frame is $\mathbf{x}_p \in \mathbb{R}^3$, is observed by multiple states and denote the set of states observing p as $\{\mathbf{S}\}_p$. For any state $\mathbf{S}_k$ in $\{\mathbf{S}\}_p$, denote the residual formed by measuring p as in the left camera image as $\mathbf{r}^V_{\xi_{k,lc},p}$ ($\xi_{k,lc}$ is the left camera pose, obtained by applying a IMU-camera transformation to ξ_k, the IMU-centered system pose):

$$\mathbf{r}^V_{\xi_{k,lc},p} = \mathbf{z}_{\xi_{k,lc},p} - h(\xi_{k,lc}, \mathbf{x}_p) \tag{2.6}$$

where $\mathbf{z}_{\xi_{k,lc},p}$ is the pixel measurement of p in the image and $h(\xi_{k,lc}, \mathbf{x}_p)$ encodes a perspective projection. Same formulation is derived for the right camera image. Iterative methods are adopted for optimizing the pose graph, and hence linearization of the above residual is required. Equation (2.7) shows the linearized residuals for landmark p.

$$\sum_{S_p} ||\mathbf{F}_{kp}\delta\xi_k + \mathbf{E}_{kp}\delta\mathbf{x}_p + \mathbf{b}_{kp}||^2 \tag{2.7}$$

where the Jacobians $\mathbf{F}_{kp}$, $\mathbf{E}_{kp}$ and the residual error $\mathbf{b}_{kp}$ are results from the linearization and normalized by $\Sigma_c^{1/2}$, the visual measurement covariance. Stacking each individual component inside the sum into a matrix we have

$$||\mathbf{r}^V_p||^2_{\Sigma_C} = ||\mathbf{F}_p\delta\xi_k + \mathbf{E}_p\delta x_p + \mathbf{b}_p||^2 \tag{2.8}$$

To avoid optimizing over $\mathbf{x}_p$, we project the residual into the null space of $\mathbf{E}_p$: Premultiply each term by $\mathbf{Q}_p \doteq \mathbf{I} - \mathbf{E}_p(\mathbf{E}_p^\top\mathbf{E}_p)^{-1}\mathbf{E}_p^\top$, an orthogonal projector of $\mathbf{E}_p$ [19]. We thus have the *Structureless Vision Factor*, for landmark p as

$$||\mathbf{r}^V_p||^2_{\Sigma_C} = ||\mathbf{Q}_p\mathbf{F}_p\delta\xi_k + \mathbf{Q}_p\mathbf{b}_p||^2 \tag{2.9}$$

2.2.3 Optimization and marginalization

Given the residuals, the pose graph optimization is a *maximum a posteriori* (MAP) problem whose optimal solution is

$$\{\mathbf{S}_t\}^*_N = \arg\min_{\{\mathbf{S}_t\}^*_N} (||\mathbf{r}_0||^2_{\Sigma_0} + \sum_{i\in w} ||\mathbf{r}^I_{i(i+1)}||^2_{\Sigma_I} + \sum_p ||\mathbf{r}^V_p||^2_{\Sigma_C}) \tag{2.10}$$

where $\{\mathbf{S}_t\}^*_N$ is the set of state variables inside the window. $\mathbf{r}_0$ and Σ_0 are prior factors and their associated covariance. Σ_I is the covariance of the IMU measurements. I use the Levenberg-Marquart optimizer to solve this nonlinear optimization problem.

The most recent N state variables are maintained inside the optimizer. Schur-Complement marginalization [45] is performed on state variables getting out of the window. Denote $\mathbf{x}_\mu$ as a set of states to be marginalized out, $\mathbf{x}_\lambda$ as the set of all states related to those by error terms, and $\mathbf{x}_\rho$ as the set of remaining states. Because of conditional independence, the marginalization step can be simplified:

$$\begin{bmatrix}\mathbf{H}_{\mu\mu} & \mathbf{H}_{\mu\lambda} \\ \mathbf{H}_{\lambda\mu} & \mathbf{H}_{\lambda\lambda}\end{bmatrix} \begin{bmatrix}\delta\mathbf{x}_\mu \\ \delta\mathbf{x}_\lambda\end{bmatrix} = \begin{bmatrix}\mathbf{b}_\mu \\ \mathbf{b}_\lambda\end{bmatrix} \tag{2.11}$$

Applying of the Schur-Complement operation yields:

$$\mathbf{H}^*_{\lambda\lambda} := \mathbf{H}_{\lambda\lambda} - \mathbf{H}_{\lambda\mu}\mathbf{H}^{-1}_{\mu\mu}\mathbf{H}_{\mu\lambda} \tag{2.12}$$

$$\mathbf{b}^*_{\lambda} := \mathbf{b}_{\lambda} - \mathbf{H}_{\lambda\mu}\mathbf{H}^{-1}_{\mu\mu}\mathbf{b}_{\mu} \tag{2.13}$$

where $\mathbf{H}^*_{\lambda\lambda}$ and $\mathbf{b}^*_{\lambda}$ are nonlinear functions of $\mathbf{x}_\lambda$ and $\mathbf{x}_\mu$. This describes a single step of the marginalization. In the VIO, marginalization is repeatedly performed and prior factors are continuely introduced. Hence, we fix the linearization point around $\mathbf{x}_0$, the value of $\mathbf{x}$ at the time of marginalization. We then introduce $\Delta x := \Phi^{-1}(\log(\bar{\mathbf{x}} \circ \mathbf{x}_0^{-1}))$ as the state update after marginalization, where $\bar{\mathbf{x}}$ is the current estimate for $\mathbf{x}$. In other words, $\mathbf{x}$ is composed as

$$\mathbf{x} = \exp\Phi(\delta\mathbf{x}) \circ \exp\Phi(\Delta\mathbf{x}) \circ \mathbf{x}_0 \tag{2.14}$$

in which

$$\bar{\mathbf{x}} = \exp\Phi(\Delta\mathbf{x}) \circ \mathbf{x}_0 \tag{2.15}$$

After introducing $\Delta\mathbf{x}$, we could approximate the right-hand side of Eq. 2.11 up to first-order

$$\mathbf{b} + \left.\frac{\partial\mathbf{b}}{\partial\Delta\mathbf{x}}\right|_{\mathbf{x}_0} \Delta\mathbf{x} = \mathbf{b} - \mathbf{H}\Delta\mathbf{x} \tag{2.16}$$

If again using the partition of variable as $\mathbf{x}_\mu$ and $\mathbf{x}_\lambda$, we have

$$\begin{bmatrix}\mathbf{b}_\mu \\ \mathbf{b}_\lambda\end{bmatrix} = \begin{bmatrix}\mathbf{b}_{\mu,0} \\ \mathbf{b}_{\lambda,0}\end{bmatrix} - \begin{bmatrix}\mathbf{H}_{\mu\mu} & \mathbf{H}_{\mu\lambda} \\ \mathbf{H}_{\lambda\mu} & \mathbf{H}_{\lambda\lambda}\end{bmatrix} \begin{bmatrix}\Delta\mathbf{x}_\mu \\ \Delta\mathbf{x}_\lambda\end{bmatrix} \tag{2.17}$$

Plug Eq. 2.17 into Eq. 2.13, we thus have

$$\mathbf{b}^*_{\lambda} := \mathbf{b}_{\lambda,0} - \mathbf{H}_{\lambda\mu}\mathbf{H}^{-1}_{\mu\mu}\mathbf{b}_{\mu,0} - \mathbf{H}^*_{\lambda\lambda}\Delta\mathbf{x}_\lambda \tag{2.18}$$

in which

$$\mathbf{b}^*_{\lambda,0} = \mathbf{b}_{\lambda,0} - \mathbf{H}_{\lambda\mu}\mathbf{H}^{-1}_{\mu\mu}\mathbf{b}_{\mu,0} \tag{2.19}$$

The marginalization procedure thus consists of applying Eq. 2.12 and Eq. 2.18. Upon marginalization, prior factors are added back to related variables (of marginalized variables) which are still inside the window, as shown in Fig. 2.1.

3 Coupling Strategy With LiDAR

3.1 Overview of the LiDAR Method

LiDAR-based methods alone demonstrate robust and accurate performance in most scenarios. In this work, I use part of LOAM [53], a state-of-the-art LiDAR method, as our LiDAR mapping system. LOAM [53] solves the SLAM problem in a division of an odometry and a mapping algorithm. The former performs odometry at a high-frequency but at low fidelity to estimate velocity of the sensor. It conducts scan to scan registration for fast computation. The latter runs at an order of magnitude lower frequency for fine matching and registration of the point cloud. It conducts point cloud dewarping and scan to map registration.

The odometry algorithm performs feature extraction and scan to scan local motion estimation. Let P_k refer to the kth point cloud, i be a point in P_k, $i \in P_k$. Let S be the set of consecutive points of i returned by the laser scanner in the same scan. Two kinds of features are extracted: edge points and planar points. The extraction is based on evaluation on the smoothness of local surface as in Eq. 3.1.

$$c = \frac{1}{|S| \cdot ||\mathbf{X}^L_{k,i}||} || \sum_{j \in S, j \neq i} (\mathbf{X}^L_{k,i} - \mathbf{X}^L_{k,j}) || \tag{3.1}$$

where $\mathbf{X}$ denotes the 3D position of the LiDAR point. Feature points are selected with the maximum c, namely, edge points, and the minimum c, namely, planar points. They form the edge point set E_k and plane point set H_k. Then, feature correspondences are found for feature point sets between scans, and scan to scan motion estimation is conducted by minimizing the overall distances of the feature points. For edge points, we use the point to line distance where the line is found as formulated by two edge identical points as in Eq. 3.2.

$$d_E = \frac{|(\tilde{\mathbf{X}}^L_{k,i} - \tilde{\mathbf{X}}^L_{k-1,j}) \times (\tilde{\mathbf{X}}^L_{k,i} - \tilde{\mathbf{X}}^L_{k-1,l})|}{|\tilde{\mathbf{X}}^L_{k-1,j} - \tilde{\mathbf{X}}^L_{k-1,l}|} \tag{3.2}$$

where $\tilde{\mathbf{X}}_{(k,i)}$, $\tilde{\mathbf{X}}_{(k-1,j)}$, and $\tilde{\mathbf{X}}_{(k-1,l)}$ are the coordinates of points i, j, and l in L_k and L_{k-1}, respectively. The latter two are retrieved edge points using the first one for formalizing this error term. For planar points, point to plane distances are used as in Eq. 3.3.

$$d_H = \frac{\left| \begin{matrix} \tilde{\mathbf{X}}^L_{k,i} - \tilde{\mathbf{X}}^L_{k-1,j} \\ ((\tilde{\mathbf{X}}^L_{k-1,j} - \tilde{\mathbf{X}}^L_{k-1,l}) \times (\tilde{\mathbf{X}}^L_{k-1,j} - \tilde{\mathbf{X}}^L_{k-1,m})) \end{matrix} \right|}{|(\tilde{\mathbf{X}}^L_{k-1,j} - \tilde{\mathbf{X}}^L_{k-1,l}) \times (\tilde{\mathbf{X}}^L_{k-1,j} - \tilde{\mathbf{X}}^L_{k-1,m})|} \tag{3.3}$$

where $\tilde{\mathbf{X}}_{(k,i)}$, $\tilde{\mathbf{X}}_{(k-1,j)}$, $\tilde{\mathbf{X}}_{(k-1,m)}$, and $\tilde{\mathbf{X}}_{(k-1,l)}$ are the coordinates of points i, j, m, and l in L_k and L_{k-1}, respectively. The latter three are retrieved planar points using the first one for formalizing this error term. In the odometry motion estimation, these two terms are jointly minimized with a small set of feature points to get an initial scan to scan motion estimate for the LiDAR mapping optimization.

The LiDAR mapping algorithm uses the scan to scan motion estimate to perform further refinements. It conducts the same optimization at 10 times lower rate but at a larger scale (use 10 times more feature points). The error terms remain the same as in the odometry part. However, feature points are dewarped into the end of the scan to account for the motion skew. Denote any time within a scan as t_i. We dewarp all points to the time of end of scan t_{k+1} based on the local motion estimate. Denote a LIDAR point at t_i as $\mathbf{P}_i$ and the dewarped itself as $\tilde{\mathbf{P}}_i$, we have

$$\tilde{\mathbf{P}}_i = (\mathbf{T}^L_{k+1})^{-1}\mathbf{T}^L_i\mathbf{P}_i \tag{3.4}$$

where $\mathbf{T}^L_{k+1}$, $\mathbf{T}^L_i$ are LIDAR frame poses from the LiDAR odometry estimate. The optimization is performed over scan points and map points (accumulated scan points from the beginning). The dewarped points are added to the map after the optimization. This refinement comes at a lower rate but is much accurate compared to the odometry. Fusing the two, LOAM outputs a refined LiDAR-rate pose estimate in real-time as the system output.

We see that the LiDAR odometry and mapping methods are relatively independent and each has its own goal, though they are similar in terms of the methodology. For the odometry, the goal is to provide a smooth and high-rate local motion estimate, which is later used for point cloud dewarping and initialization of the scan to map registration. For the mapping, the goal is to reliably register new coming scans to the existing map. It should be accurate and robust enough to correct the local drift and built a consistent map.

3.2 Loose Coupling Strategy

From the relative independence of the odometry and mapping algorithm, one could think of substitute either parts to improve the overall performance of the algorithm. From experiments, I see that the LiDAR odometry is relatively inaccurate. Sometimes even the local drift could be pretty large. However, the LiDAR mapping is rather robust and accurate. With a VIO in hand, I hence substitute the LiDAR odometry with the VIO. This is shown in the system diagram Fig. 1.1.

To use the VIO as the odometry source, the LiDAR mapping system uses high frequency IMU-rate VIO poses as the motion prior to perform LiDAR points dewarping and scan to map registration. To ensure that the local smoothness and consistency are preserved , I generate a smoothed version of the VIO poses by

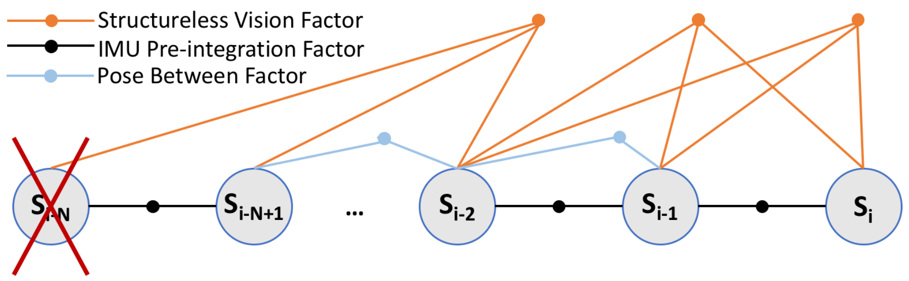


Figure 3.1: Fixed-lag pose graph formulation in the VIO with LiDAR feedback pose between factor. State variables being optimized are circled, where i stands for the current state and N is the window size. The state to be marginalized is crossed. The light blue edge and dot indicate the pose between factor from LiDAR feedback.

integrating the IMU measurements with the corrected biases and velocities. Altough this estimate is sub-optimal, experiments show that when the optimization problem is well constrained, the estimate is reasonably good as a local pose source. Feature works from the LiDAR odometry are still needed. Denote a scan χ as the point cloud obtained from one complete LiDAR rotation. Geometric features including points on sharp edges and planar surfaces are extracted from χ as in Sec. 3.1. Then, all feature points are dewarped. The registration is then based on dewarped feature points from current scan to the map (all previous feature points), solved as an optimization problem by minimizing Euclidean distance residuals formed by the feature points as in Eq. 3.2 and 3.3.

3.3 Tight Coupling Strategy

Though the coupling strategy in Sec. 3.2 integrates the VIO and LiDAR mapping, it lacks the capability of correcting the odomety (specifically the IMU biases and velocity estimate) using the LiDAR mapping results, which is referred as feedback from LiDAR in [54]. The disadvantage is obvious: the odoemtry estimate could suffer once the robot is in visually challenging environments, which in turn affects the performance of the LiDAR mapping system. Hence, I investigate a feedback mechanism from LiDAR mapping to VIO, which is naturally supported by the pose graph formulation.

Consider two pose estimates from the LiDAR mapping system L_{t-1} and L_t at time $t-1$ and t. From the odometry perspective, they are of smaller drifts compared with the VIO's estimates. Hence, I add another constrain in the VIO's pose graph to model this relative information, namely, a *Pose Between Factor*. Using $\mathbf{r}^F$ to denote the assoicated residual:

$$\mathbf{r}^F = (L_{t-1}^{-1} L_t)^{-1} (S_{t-1}^{-1} S_t) \tag{3.5}$$

in which $(L_{t-1}^{-1} L_t)$ is the measurement of the local relative motion with an associated noise model. S_{t-1} and S_t are the current estimate for the two poses inside the state

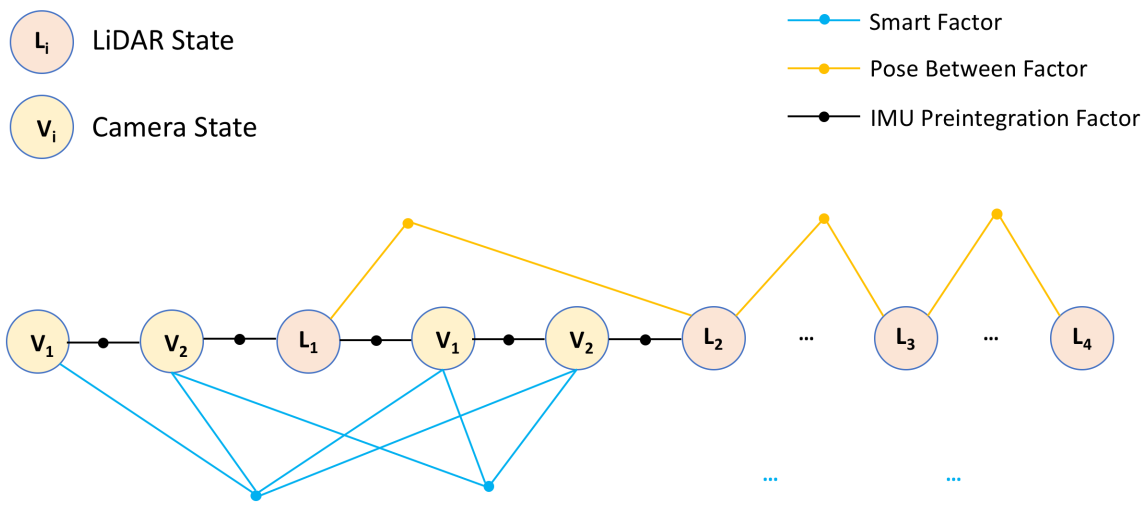


Figure 3.2: A more practical formulation of the pose graph inside VIO with LiDAR feedback. LiDAR states and camera states have the same state variables but are different in terms of the timestamp. Pose between factors are formulated on LiDAR states. Camera states follow the ones in the original VIO pose graph as in Fig. 2.1.

variables. Note that the residuals are formulated on manifold using the exponential map. The rotation part is minimized towards identity and the translation part is minimized towards zero. The revised pose graph in the VIO is shown in Fig. 3.1. Some practical issues are also worth to discuss. First, LiDAR mapping takes time to finish. Hence, the pose between factors are always added onto the states that are back in the window. Second, in Fig. 3.1 the pose between factors are added to the same state variables as of the keyframe states in the original VIO setup. This makes assumption to the sensor platform: LiDAR scans are received approximately at the same time of the keyframe states. However, LiDAR scans and images are very likely to come at a different rate. A more practical or general model would be the one shown in Fig. 3.2. LiDAR states are created separately and IMU pre-integration factors are re-generated to account for the insertion of the LiDAR state variables. In this work, experiments are carried out with the model in Fig. 3.1 on a well synchronized dataset to illustrate the idea.

4 LiDAR Enhanced Visual Loop Closure

Loop closure is critical to any SLAM system as long term operation introduces drift. The objective of loop closure is to eliminate drift by performing a global pose graph optimization which incorporates loop constraints and relative transformation information from LIDAR mapping. To better assist LIDAR mapping, the corrected LIDAR pose is sent back in real-time so that feature points from new scans are registered to the revisited map. We propose adding ICP alignment in addition to visual Bag-of-Words [20] loop detection and PnP loop constraint formulation. The system uses iSAM2 [25], an incremental solver, to optimize the global pose graph, achieving real-time performance.

4.1 Loop Detection

We use the bag-of-words method [20] for initial loop detection. The bag-of-words model is trained beforehand using images from alike environments. During operations, keyframes with their pose estimates are added to the database at its first appearance. At the same time, it retrieves for the most similar keyframe from the bag-of-words database to see if there is a potential loop. To prevent false loop detection, we restrict candidates within a certain time threshold. If retrieved, the system matches feature descriptors of the left image with the loop candidates to further filter out false positives.

4.2 Loop Constraint Generation

The system first obtains visual loop constraint as an initial estimate. Since we use a structureless formulation for visual landmarks, triangulation on all the stereo matched features in the loop candidate is performed to obtain their 3D location. Their associations to current keyframe are given by descriptor match. The visual loop constraint is then evaluated using EPNP [**?**]. For each keyframe, we also store the associated LiDAR key scans using timestamp. To improve the accuracy of the visual loop constraint, we use ICP alignment on the feature points of the corresponding LIDAR key scans. With a bad initialization or a larger point count, ICP takes longer to converge and consumes more computation resources. However, the visual loop constraint provides a good initialization point and the ICP only uses sparse feature points (Sect. 3.1), which makes it converge fast.

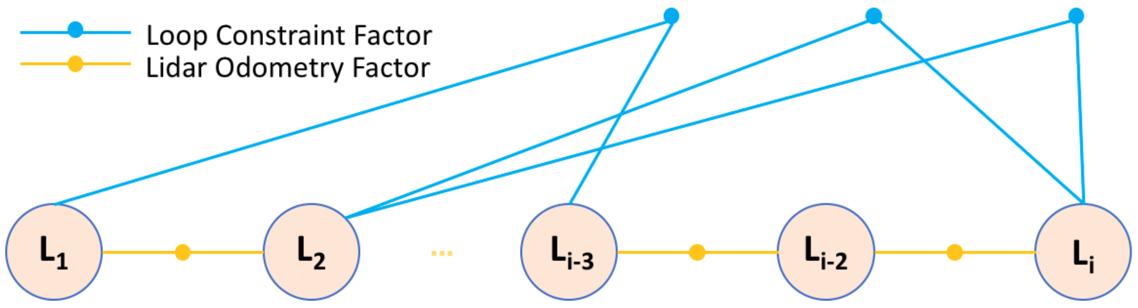


Figure 4.1: The global pose graph consists of the *LiDAR Odometry Factor* and the *Loop Constraint Factor*. i stands for the most recent LiDAR scan.

4.3 Global Pose Graph Optimization

The graph representation of the global pose graph is shown in Fig. 4.1. It contains all the available LiDAR mapping poses as variables, constrained by the *LiDAR Odometry Factor* and the *Loop Constraint Factor*, both having the same residual form as the *Pose Between Factor* shown in Eq. 3.5. The residual is expressed in 6 DoF minimum form in the optimization. For the *LiDAR Odometry Factor*, the two poses to form the pose between factor are all of the consecutive poses. For the *Loop Constraint Factor*, the two poses are the detected twos to formulate a loop. To further improve the efficiency and realize real-time update performance, we adopt iSAM2 [25] to incrementally optimize the pose graph.

4.4 Re-localization

Once a loop closure candidate is detected, LIDAR mapping buffers the feature points (without registering them to the map) until it receives loop correction. The loop correction contains globally optimized trajectory. LIDAR mapping updates its map, adds the buffered feature points to the map and then resumes its operation. We can afford to update the map in real-time because (a) loop closure has a real-time performance (b) the sparse feature map does not take much memory, and (c) scan to map registration is fast enough to catch up with the LIDAR data rate.

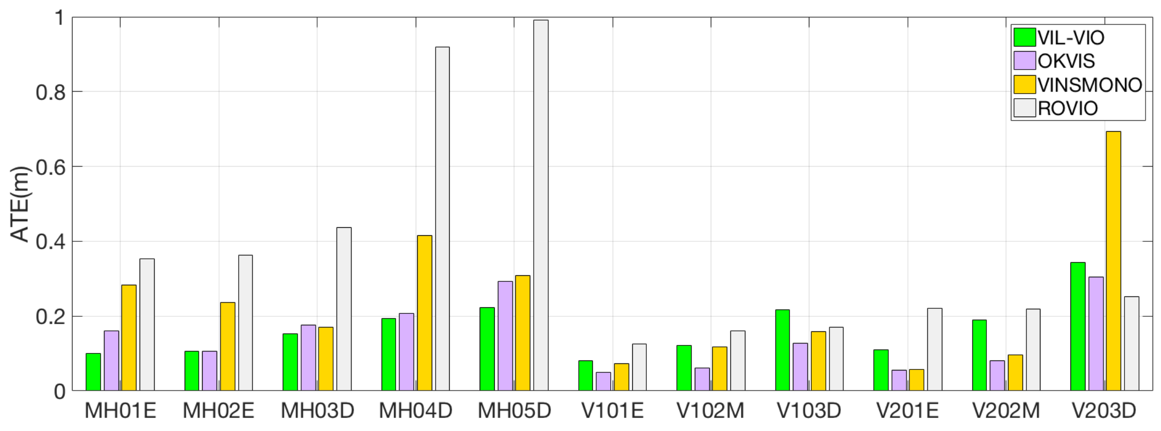


Figure 5.1: Root Mean Square Error (RMSE) of ATE for EoRoC MAV Dataset. Methods are colored differently as in the legend for better visualization.

5 Experimental Results

The software pipeline is implemented in C++ with ROS communication interface. We use GTSAM library [12] to build the fixed-lag smoother in the VIO. For loop closure, we use LibPointMatcher [39] to do ICP on point clunds, DBoW3 [2] to build the visual dictionary, and iSAM2 [25] implementation in GTSAM [12] to conduct global pose graph optimization. I have used data from both public available datasets and custom collected dataset to evaluate the performance of the algorithm. Specifically, I use the EuRoC MAV dataset [6] to evaluate the stereo VIO implementation, a custom dataset (Autel dataset) to evaluate the robustness of the complete pipeline without LiDAR feedback, and the KITTI dataset [21] to evaluate the idea of tightly coupling LiDAR and VIO.

5.1 EoRoC MAV Dataset

I evaluate the stereo VIO implementation (VIL-VIO) using the EuRoC MAV dataset [6] in terms of the Absolute Trajectory Error (ATE) as in [46]. This dataset consists of 20Hz stereo images and 200Hz IMU information recorded by a VI sensor [38] mounted on a MAV. There are 11 sequences varying in motion dynamics, lighting conditions and image qualities. The difficult ones in V1 and V2 present aggressive motion, motion blur, and illumination changes, which all challenge a state estimator.

Table. 5.1 and Fig. 5.1 show the comparison results between VIL-VIO and some representative state-of-the-art methods. OKVIS [29] and VINS-MONO [40] are fixed-lag VIO systems and ROVIO [3] is a filtering-based approach. In Table. 5.1 and Fig. 5.1, a sequence is named in the first four letters and the difficulty level is encoded in the last letter (E:easy, M:medium, D:difficult). Results for VIL-VIO are deterministic, obtained in real-time on a desktop with 3.60GHz i7-4790 CPU. Results for the other methods are the better ones from experiments in [23] and [47]. The best results are bold in the table. VIL-VIO achieves the best in five out of eleven sequences and shows competitive results on the others. It succeeds all the difficult ones, verifying the capability to handle aggressive motion, illumination changes, motion blur and textureless regions reasonably well. However, we see that OKVIS outperforms VIL-VIO in six sequences out of the eleven. Though the differences are small, we should note that OKVIS has a more complicated marginalization strategy and it has a frontend that is more blended with the backend optimizer for feature selection. These give rooms for VIL-VIO to improve. If directly compare with VINS-MONO, we see that VINS-MONO reaches better performance in five of the eleven sequences. However, it could have pretty bad performance in the difficult ones wheareas VIL-VIO has a more uniformed performance in all difficult levels. The reason might be the scale estimation in VINS-MONO is not robust enough to handle all scenarios, especially in those challenging sequences.

Table 5.1: RMSE OF ATE (METER) ON THE EuRoC MAV DATASET [6]

Sequence	VIL-VIO	OKVIS	VINS-MONO	ROVIO
MH-01-E	**0.100**	0.160	0.284	0.354
MH-02-E	**0.106**	0.106	0.237	0.362
MH-03-M	**0.153**	0.176	0.171	0.436
MH-04-D	**0.194**	0.208	0.416	0.919
MH-05-D	**0.223**	0.292	0.308	0.991
V1-01-E	0.080	**0.050**	0.072	0.125
V1-02-M	0.121	**0.061**	0.118	0.160
V1-03-D	0.217	**0.127**	0.159	0.170
V2-01-E	0.109	**0.055**	0.058	0.220
V2-02-M	0.189	**0.081**	0.097	0.218
V2-03-D	0.343	0.305	0.693	**0.252**

There are plenty of rooms for VIL-VIO to improve. Firstly, VIL-VIO is a stereo implementation but not using the stereo disparity map. During operation, usually there are not more than a hundred feature matches currently. Hence, a better frontend would definitely boost its performance. For example, one could integrate disparity map or depth map from the sensor or some other software pipeline (e.g. deep learning approaches). This gives more clues and information for feature selection, feature match generation, and early outlier rejection. Secondly, a more thoughtful marginalization that separates recent frame and keyframe could be implemented as in [29, 40]. This helps maintain large parallax between keyframes to avoid degeneracy and better constrain the biases estimation using the recent frames.

Table 5.2: FDE (%) and MRE (m) TEST RESULTS

Test	Total Length	FDE		MRE	
		VIL-SLAM	LOAM	VIL-SLAM	LOAM
Highbay	118	**0.08**	0.56	**0.08**	0.22
Hallway	103	**0.61**	0.91	**0.10**	0.27
Tunnel	85	**1.86**	-[2]	×	×
Huge Loop	318	**0.01**	-	**0.22**	0.36
Outdoor	528	0.02	0.02	×	×

5.2 Autel Dataset

We built a platform with two mega-pixel cameras, a 16 scan-line LiDAR, an IMU (400Hz), and a 4GHz computer (with 4 physical cores). Synchronizing the time between the sensors and computer is critical and directly affects the quality of any SLAM system. We built a custom microcontroller based time synchronization circuit that synchronizes the cameras, LIDAR, IMU and computer by simulating GPS signals (PPS and NMEA stream). We have collected data from various challenging environments with this platform, aiming to fully evaluate VIL-SLAM. In comparison, I also show results of LOAM [53], the best real-time LiDAR based system[1], on these test sequences.

I present results from five representative environments including featureless hallways, cluttered highbays, tunnels, and outdoor environments. The data collection started and ended at the same point for all these sequences. Odometry (LiDAR mapping pose) is evaluated based on the final drift error (FDE). Mapping results are evaluated in terms of mean registration error (MRE) using Faro scans as ground truth: I first align the map with the model (Faro scans), and then compute the Euclidean distance between a map point and its closest point in the model [1]. The odometry FDE and mapping results are shown in Table 5.2 with the better ones in bold. The trajectories and cross-sectioned maps are shown in Fig. 5.2, 5.3, 5.4, 5.5, 5.6. The map comparisons are shown in Fig. 5.7.

The *highbay* is an indoor warehouse which is open, structured, and rich in features. However, frequent structural occlusions could be a challenge for the visual frontend and the LiDAR feature extraction part. Both VIL-SLAM and LOAM handle this environment pretty well. For VIL-SLAM, LiDAR mapping module registers most of its scan to map, largely reducing the odometry error. Loop closure recognizes the starting position and closes the loop. The map is generated using the globally refined

[1]This is the best implementation of LOAM I could find online https://github.com/laboshinl/loam_velodyne

[2]"-" indicates not finished. "×" indicates missing data.

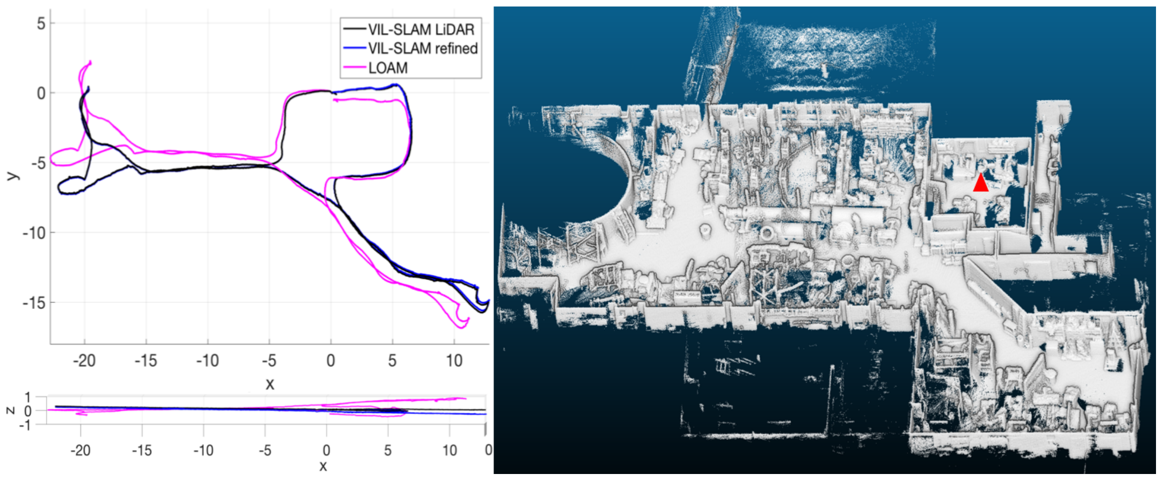


Figure 5.2: Trajectory and mapping results for the *highbay* test. Trajectories from VIL-SLAM and LOAM are shown on the left and cross-sectioned maps generated by VIL-SLAM are shown on the right. Start(end) position is labeled with red triangle in the map and is the origin in the plot. Loop closure is triggered and the globally refined trajectory is shown in blue.

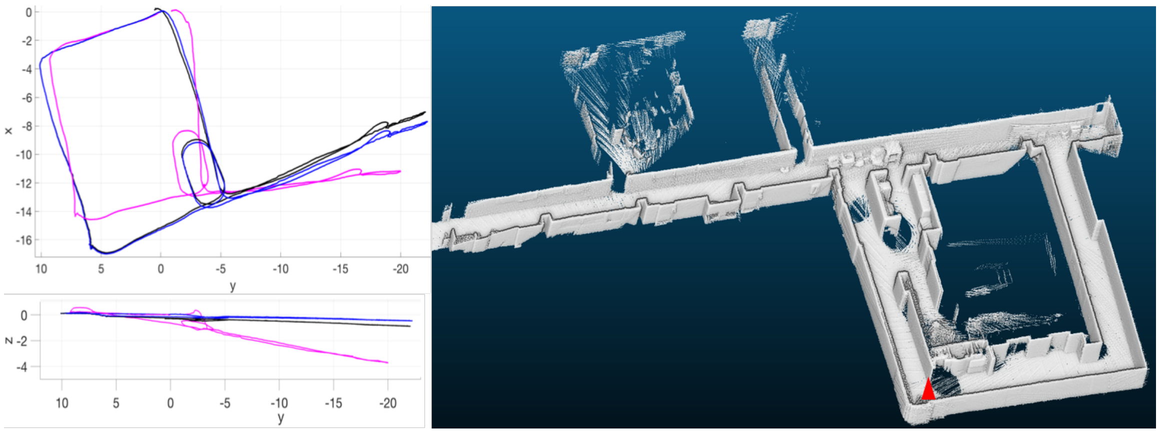


Figure 5.3: Trajectory and mapping results for the *hallway* test.

poses, with the majority of map errors below 0.15m.

The *hallway* and *tunnel* tests are challenging environments because of lack of visual features and the degeneracy issue along traversal direction for LiDAR. LOAM accumulates large error in the hallway, and fails the tunnel test mainly due to the degeneracy issue. Aided by the stereo VIO module (VIL-VIO), VIL-SLAM succeeds both tests. In the *hallway* test, the visual frontend returns fewer reliable measurements because of the featureless walls, under-constraining the VIO. This corrupts the map as observed by wall misalignment, which is later corrected by loop closure as shown in Fig. 5.8(c-d). Loop closure detects the loop twice when approaching the endpoint, lowering FDE to 0.05% and generating a refined map. In the *tunnel* test, because of the degeneracy issue, VIL-SLAM struggles as well and accumulates some

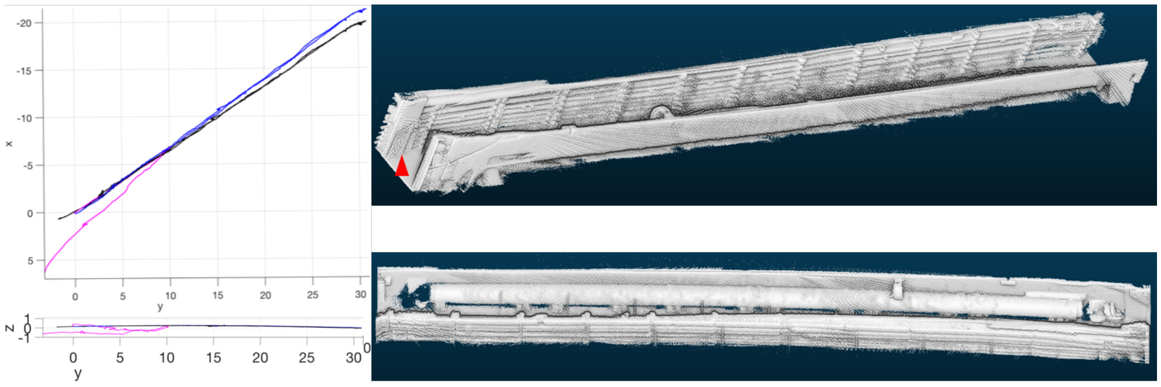


Figure 5.4: Trajectory and mapping results for the *tunnel* test.

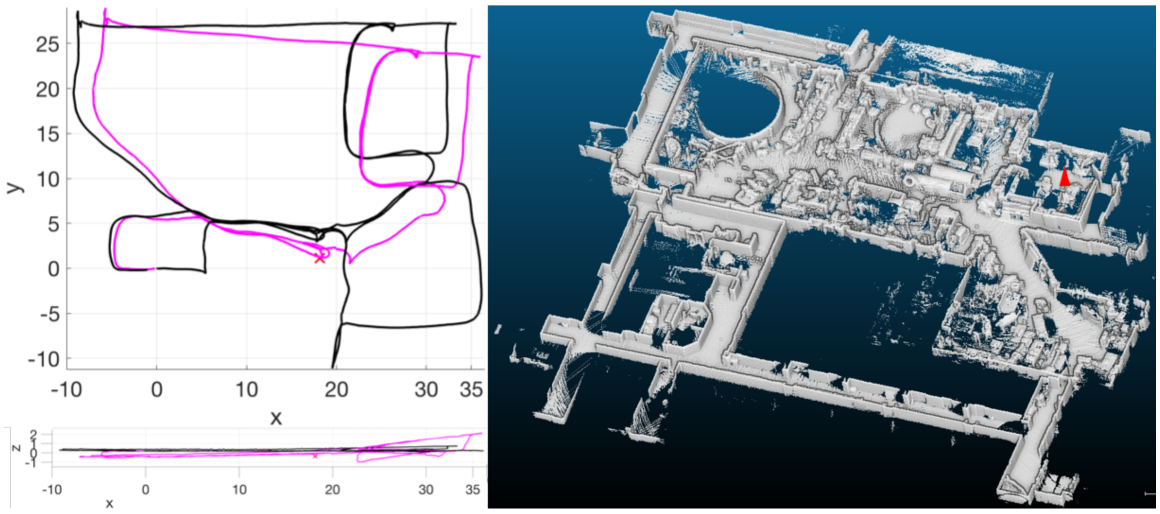


Figure 5.5: Trajectory and mapping results for the *huge loop* test.

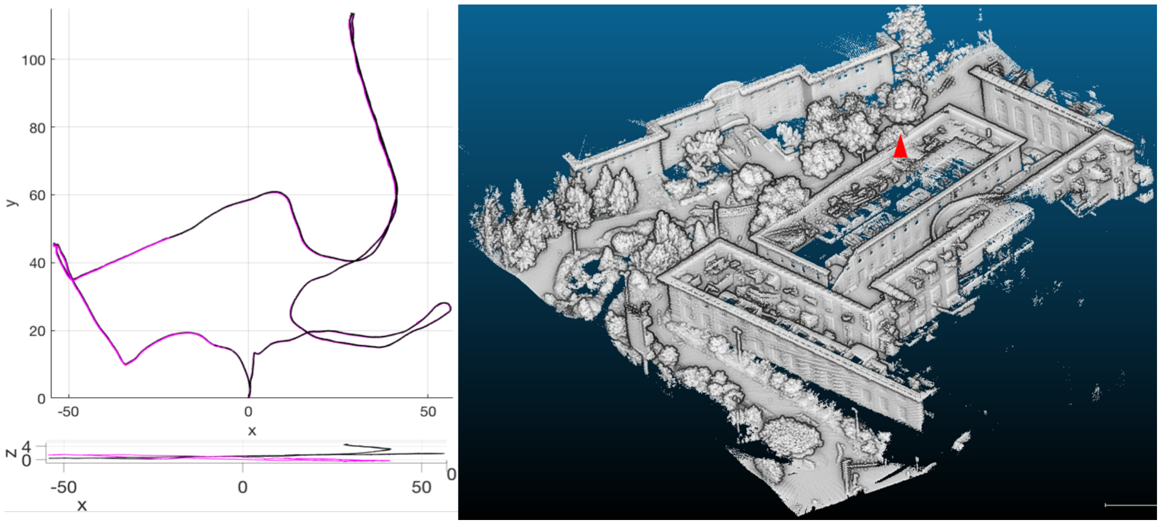
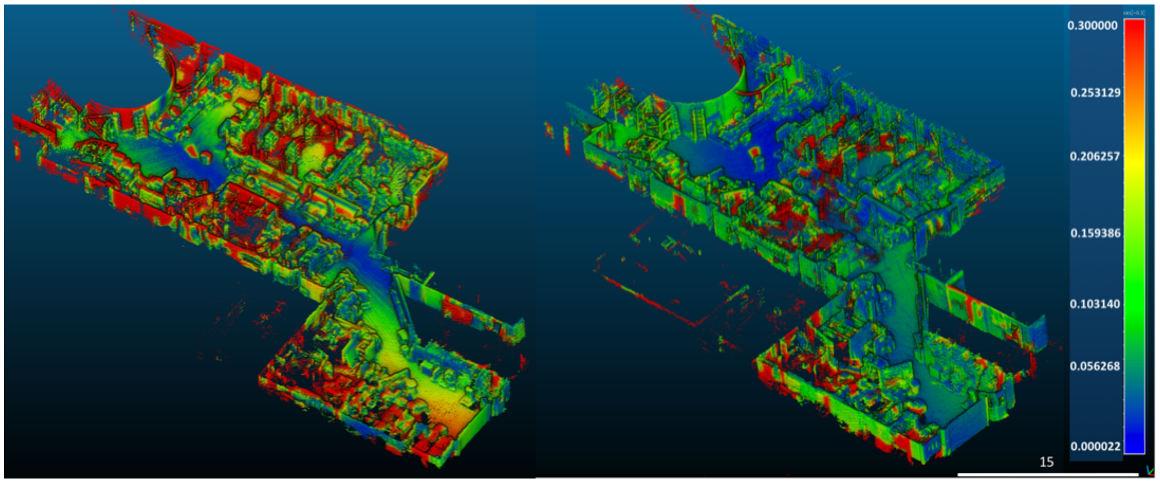
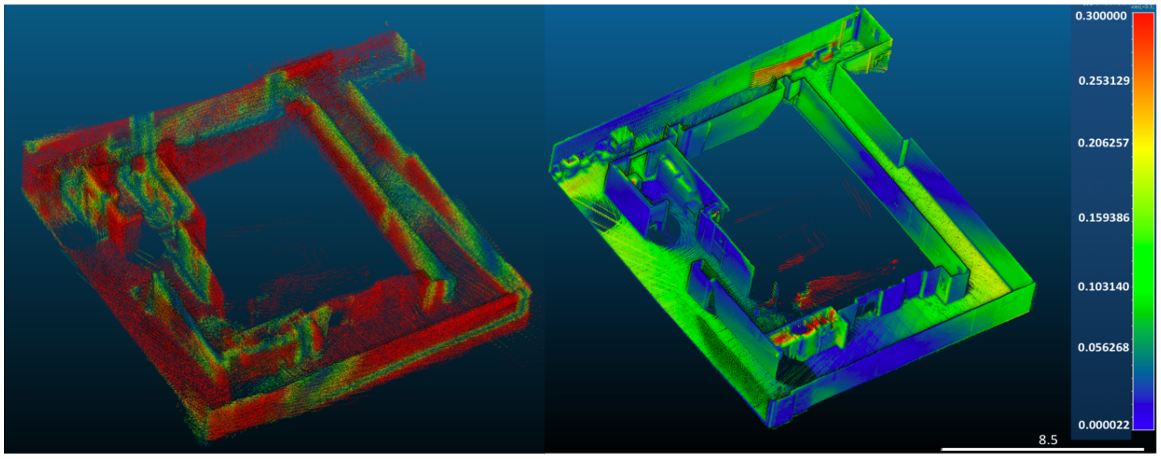


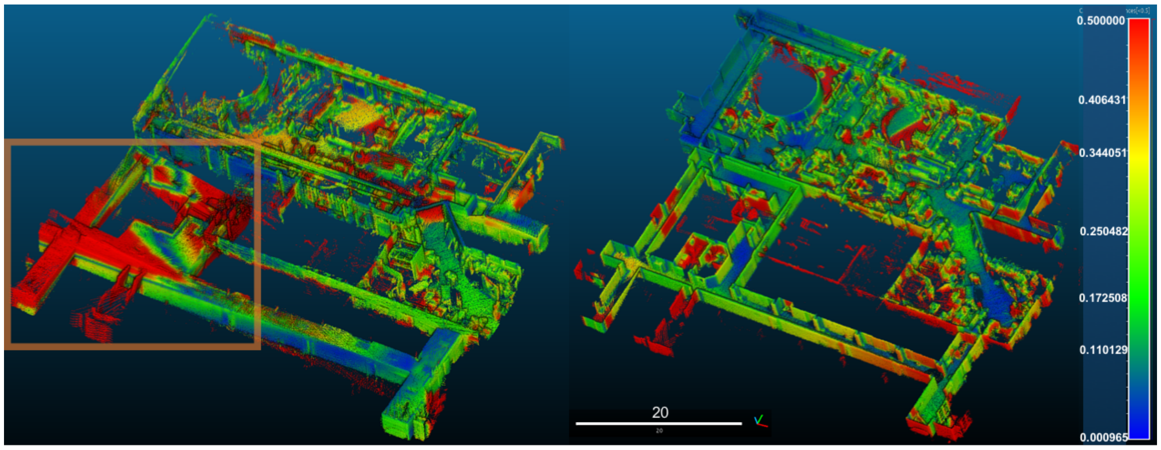
Figure 5.6: Trajectory and mapping results for the *outdoor* test.



(a) Highbay



(b) Hallway



(c) Huge Loop

Figure 5.7: Map registration error of VIL-SLAM (right) and LOAM (left) comparing to the model. Errors above 0.3m are colored red for (a-b) and 0.5m for (c). Discontinuous red regions inside the blue and green are due to lack of the model caused by occlusions of the Faro scans.

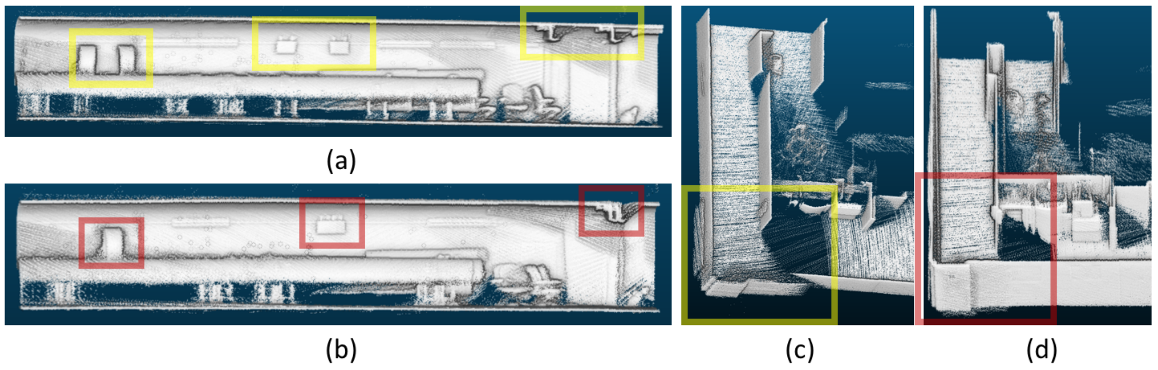


Figure 5.8: (a) Map of the tunnel stitched using LIDAR mapping poses. (b) Map of the tunnel stitched using globally refined poses. Double image in (a) is mostly eliminated but not fully, because only one loop constraint is generated, not enough for a full correction. (c) Map of the hallway stitched using LIDAR mapping poses. (d) Map of the hallway stitched using globally refined poses. Double image in (c) is mostly eliminated. Walls are aligned with two loop constraints.

error in the traversal direction. However, loop closure detects the loop at about 3m from the end point, lowering the FDE down to 0.08% and correcting the map as shown in Fig. 5.8(a-b).

The *huge loop* test features challenges from both *hallway* and *highbay* environments. In addition, we end the trajectory by re-entering the highbay after traversing along a long narrow corridor. LOAM fails this test after re-entering the highbay, at the place labeled by a red cross in Fig. 5.5. This is because it fails to register new scans to the original *highbay* map caused by a large error in z-direction accumulated in the corridor. VIL-SLAM succeeds in this test. Without loop closure being triggered, it achieves 0.01% FDE in odometry. VIL-SLAM is robust and achieves this result by successfully registering new scans to the original *highbay* map at re-entry. The map generated with the odometry estimate of VIL-SLAM is compared with the map generated with LOAM before its failure. The boxed region is where LOAM accumulates errors leading to its failure.

The *outdoor* test features an outdoor trajectory which is 546m long and includes a gentle slope. Pedestrians and cars were observed which served as potential outliers. VIL-SLAM and LOAM have comparable results along the xy-plane. However, LOAM fails to capture the changes in the z-direction. The inaccuracy in z of LOAM is also observed in the previous tests.

Overall, VIL-SLAM generates more accurate mapping results and lower FDE compare to LOAM when they both finish. Also, VIL-SLAM succeeds the more challenging environments where LOAM fails with qualitatively good mapping and odometry results.

5.3 KITTI Odometry Dataset

I use training sequences from KITTI odometry dataset [21] to validate the LiDAR feedback mechanism. The dataset consists of driving sequences collected with a platform mounted on top of a vehicle, featuring environments including country, urban, and highway. The dataset provides data from high resolution color and grayscale stereo cameras, a 64-line Velodyne 3D laser scanner, and a high-precision GPS/IMU inertial navigation system. Moreover, data from different sensors is well synchronized and is well suited for the pose graph formulation in 3. However, there are sequences with lost of IMU messages occasionally, which VIL-SLAM could not deal with at the point. Hence, only results from sequences 05 to 10 are shown below. Also note that the VIO system that achieve these results is of a monocular version that I implemented. The reason is while LiDAR could recover the scale, the visual frontend could get rid of the stereo matching phase which eliminates lots of visual observations. In this monocular version, I use KLT feature tracker [4] to generate temporal constraints. I adopt the official evaluation metric from KITTI: From all sequences, it computes translational and rotational errors for all possible sub-sequences of length (100,...,800) meters. Errors are measured in percentage (for translation) and in degrees per meter (for rotation).

Results for the mean translational errors are shown Table 5.3. Qualitative results are shown in Fig. 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, and 5.15. In the table, TC indicates tight-coupling and LC indicates loose-coupling. Results of VIL-SLAM (monocular version) running in both loose and tight coupling mode are shown. In comparison, results of LOAM [53] are shown. Note that for sequence 05, results of VIL-SLAM are for the first half of the trajectory as IMU messages are lost in the middle. Comparing TC and LC, we see results for TC are all better. This validates the initial expectation, as the LiDAR feedback mechanism could help constrain the scales in a monocular VIO, which in turn yields much better local odometry estimate.

Table 5.3: MEAN TRANSLATIONAL ERROR (%) ON THE KITTI ODOMETRY DATASET

Sequence	Environments	VIL-SLAM (TC)	VIL-SLAM (LC)	LOAM
05	Urban	**0.48**	0.51	0.57
06	Urban	0.89	0.93	**0.65**
07	Urban	**0.52**	0.73	0.63
08	Urban+Country	**1.01**	3.75	1.12
09	Urban+Country	0.79	0.878	**0.77**
10	Urban+Country	1.53	1.49	**0.79**

However, comparing with LOAM, the tight coupling VIL-SLAM only achieved better

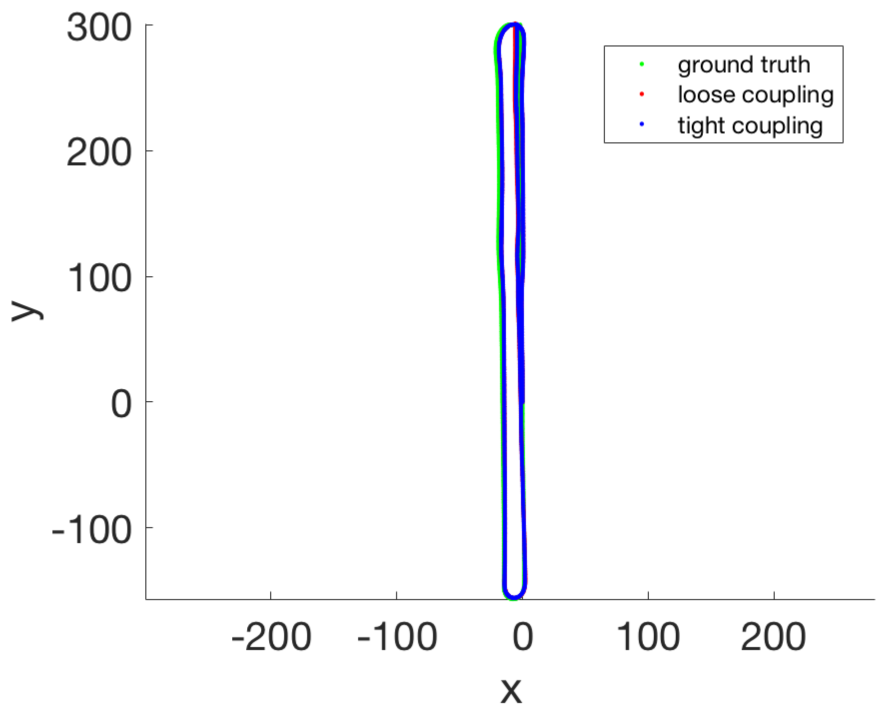


Figure 5.9: Trajectory results for KITTI sequences 06.

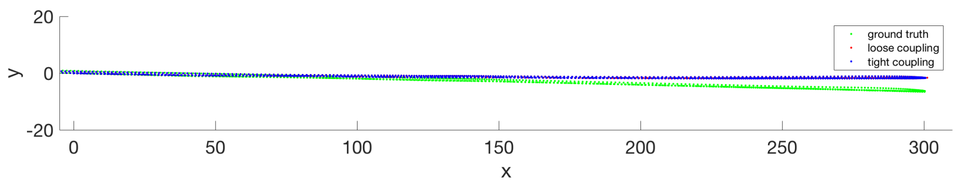


Figure 5.10: Trajectory results for KITTI sequences 06 (side).

results in 3 of the 6 sequences. In sequence 06 and 10, the results are much worse in comparison, and in sequence 09, the two methods achieves relatively similar performance. I believe this inaccuracy is because of two major disadvantages of the current implementation. Firstly, no explicit scale estimation is performed in the current monocular VIO implementation. The odometry estimate is up to scale and hence leads to unstable and inaccurate results, especially in the loose coupling case. In the tight coupling case, though LiDAR feedback mechanism corrects the scale ambiguity, an explicit scale estimate in the VIO is much more self-contained and brings more robustness to the estimation process. Secondly, the current LiDAR implementation deals with z-direction estimation poorly in certain environments, and the estimate drifts along z-direction more in comparison with x and y directions.

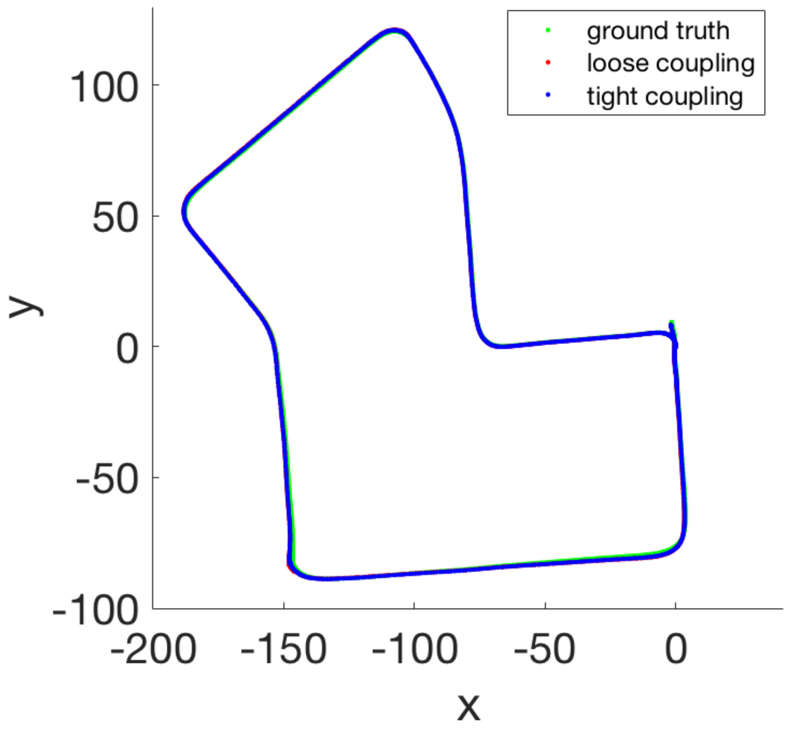


Figure 5.11: Trajectory results for KITTI sequences 07.

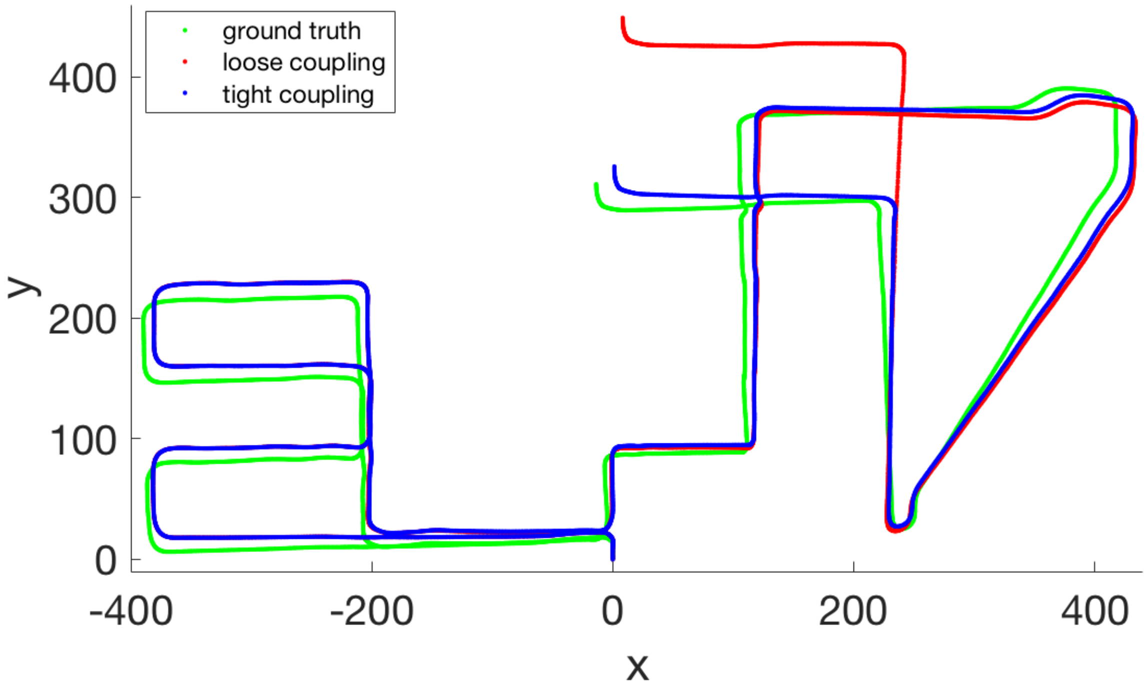


Figure 5.12: Trajectory results for KITTI sequences 08.

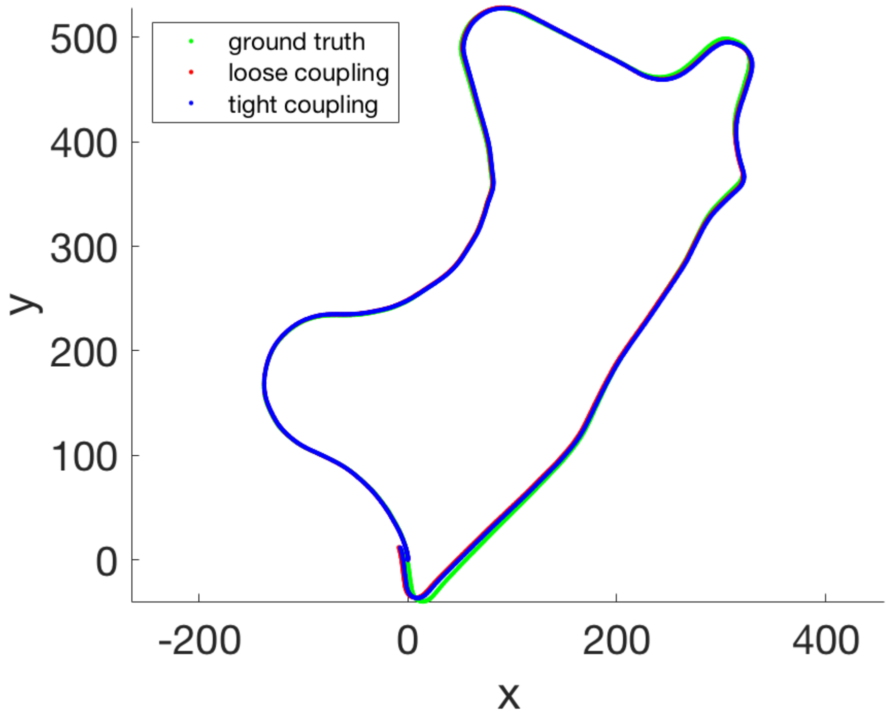


Figure 5.13: Trajectory results for KITTI sequences 09.

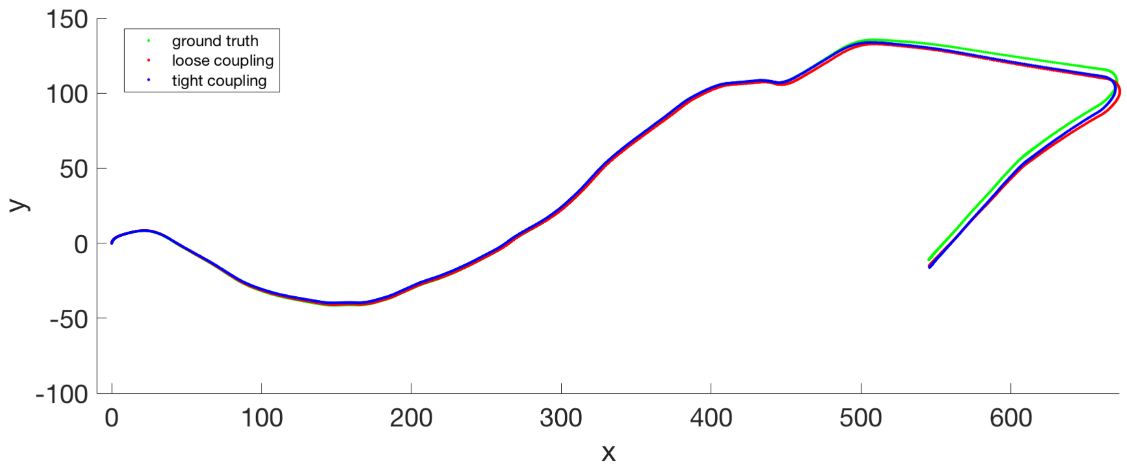


Figure 5.14: Trajectory results for KITTI sequences 10.

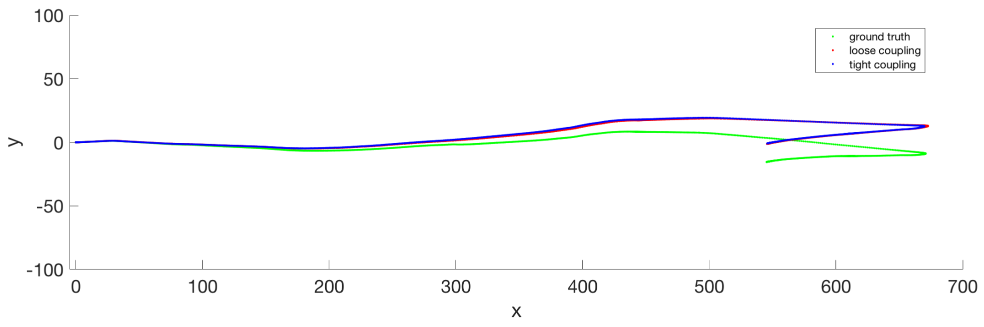


Figure 5.15: Trajectory results for KITTI sequences 10 (side).

This can be seen from both Fig. 5.10 and 5.15: The majority of the drift is along z-dimension and is consistent in both loose coupling and tight coupling cases. To remedy the z-drift issue, I believe adjusting weights for different losses and adding more penalty terms such as global plane and line constraints should help. The results for sequence 09 are very close in comparison. From the figure, We see that there are drifts in x and y direction at the end part of the trajectory for VIL-SLAM, and the one for tight coupling case is a bit better.

Overall, the experiments demonstrate that adding LiDAR feedback mechanism to the VIO is beneficial for both fixing the scale and lowering the drift. The results should be improved if a stereo VIO is used in which case there is no scale ambiguity, and if the LiDAR mapping system is improved to be more robust to estimation along z-direction.

6 Conclusion

In this thesis, I have explored a multi-sense approach for improving the robustness of a SLAM pipeline. I investigated the fusion of LiDAR, camera and IMU for tackling the real-time state estimation and mapping problem in a factor graph formulation. The method consist of a stereo VIO, a LiDAR mapping system, and a LiDAR enhanced visual loop closure system. It runs two factor graph optimization in real-time: a fixed-lag smoothing in the stereo VIO and a global pose graph optimized incrementally. The complete pipeline, VIL-SLAM, is able to generate loop-closure corrected 6-DOF poses and associated sparse LiDAR point cloud in real-time, and dense 1cm voxel-size maps near real-time. VIL-SLAM has been evaluated successfully on public dataset (EuRoC MAV dataset) and a custom dataset (the Autel dataset) collected with a built sensor suite. The pipeline demonstrates improved accuracy and robustness in challenging environments, where state-of-the-art LiDAR methods fail easily. Then, I further discussed loose and tight coupling strategies between the VIO and the LiDAR mapping system, and introduced the LiDAR feedback mechanism to help reduce drifts in the VIO local odometry estimate. This mechanism is successfully validated with the KITTI odometry dataset.

Several potential directions could be investigated for improvements. Firstly, visual frontend needs a more careful examination, both in terms of feature generation, association, and outlier rejection. Information such as depth map in a stereo case and LiDAR point cloud projected to image plane are valuable to be included in this pipeline. Moreover, deep learning based temporal and spatial feature association methods are worth an investigation. In terms of outlier rejection, the frontend should get more integrated with the backend, and use the current estimate as one cue for eliminating outliers. Secondly, the LiDAR feedback mechanism is now evaluated with a monocular VIO, in which case the scale ambiguity is not explicitly dealt with. Thus, either projecting LiDAR point cloud to image space and use it as frontend information in the monocular VIO or using a stereo VIO could be investigated to boost the performance. Finally, the LiDAR feedback mechanism implementation should be revised for the formulation that cooperates more practical sensor configurations as in Fig. 3.2, after which a more comprehensive evaluation should be performed.

Bibliography

[1] Cloud-to-Cloud Distance cloudcompare. `https://www.cloudcompare.org/doc/wiki/index.php?title=Cloud-to-Cloud_Distance`, 2015.

[2] DBoW3 dbow3. `https://github.com/rmsalinas/DBoW2`, 2017.

[3] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct ekf-based approach. pages 298–304, Sept 2015.

[4] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. 1, 01 2000.

[5] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic data association for semantic slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1722–1729, May 2017.

[6] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, Dec 2016.

[8] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert. Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4290–4297, May 2014.

[9] S. Ceriani, C. Snchez, P. Taddei, E. Wolfart, and V. Sequeira. Pose interpolation slam for large maps using moving 3d sensors. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 750–757, Sept 2015.

[10] Winston Churchill and Paul Newman. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, 2013.

[11] Igor Cvii, Josip esi, Ivan Markovi, and Ivan Petrovi. Soft-slam: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles. *Journal of Field Robotics*, 35(4):578–595.

[12] Frank Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical report, Georgia Tech, September 2012.

[13] Jeffrey A. Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. 2018.

[14] Jean-Emmanuel Deschaud. Imls-slam: scan-to-model matching based on 3d data. *CoRR*, abs/1802.08633, 2018.

[15] D. Droeschel, J. Stckler, and S. Behnke. Local multi-resolution representation for 6d motion estimation and mapping with a continuously rotating 3d laser scanner. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5221–5226, May 2014.

[16] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, March 2018.

[17] J. Engel, J. Sturm, and D. Cremers. Camera-based navigation of a low-cost quadrocopter. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2815–2821, Oct 2012.

[18] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, May 2014.

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

[20] Dorian G'alvez-L'opez and J. D. Tard'os. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[21] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[22] Joel A Hesch, Dimitrios G Kottas, Sean L Bowman, and Stergios I Roumeliotis. Camera-imu-based localization: Observability analysis and consistency improvement. *The International Journal of Robotics Research*, 33(1):182–201, 2014.

[23] J. Hsiung, M. Hsiao, E. Westman, R. Valencia, and M. Kaess. Information sparsification in visual-inertial odometry. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Madrid, Spain, October 2018. To appear.

[24] Vadim Indelman, Stephen Williams, Michael Kaess, and Frank Dellaert. Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems*, 61:721–738, 2013.

[25] 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. In *2011 IEEE International Conference on Robotics and Automation*, pages 3281–3288, May 2011.

[26] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, Nov 2007.

[27] Dimitrios G. Kottas, Joel A. Hesch, Sean L. Bowman, and Stergios I. Roumeliotis. *On the Consistency of Vision-Aided Inertial Navigation*, pages 303–317. Springer International Publishing, Heidelberg, 2013.

[28] Kin Leong Ho and Paul Newman. Loop closure detection in slam by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 54:740–749, 09 2006.

[29] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. 34, 02 2014.

[30] S. Lowry, N. Snderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, Feb 2016.

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

[32] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3923–3929, Nov 2013.

[33] M. J. Milford and G. F. Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE International Conference on Robotics and Automation*, pages 1643–1649, May 2012.

[34] Maher Moakher. Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications*, 24, 04 2002.

[35] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, April 2007.

[36] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, April 2007.

[37] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, Nov 2011.

[38] Janosch Nikolic, Jörn Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul Timothy Furgale, and Roland Siegwart. A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 431–437, 2014.

[39] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, 34(3):133–148, February 2013.

[40] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, Aug 2018.

[41] Meixiang Quan, Songhao Piao, Minglang Tan, and Shi-Sheng Huang. Map-based visual-inertial monocular slam using inertial assisted kalman filter. 09 2017.

[42] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011.

[43] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, Dec 2011.

[44] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994.

[45] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608.

[46] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, Oct 2012.

[47] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3(2):965–972, April 2018.

[48] V. Usenko, J. Engel, J. Stckler, and D. Cremers. Direct visual-inertial odometry with stereo cameras. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1885–1892, May 2016.

[49] M. Velas, M. Spanel, and A. Herout. Collar line segments for fast odometry estimation from velodyne point clouds. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4486–4495, May 2016.

[50] C. Wei, T. Wu, and H. Fu. Plain-to-plain scan registration based on geometric distributions of points. In *2015 IEEE International Conference on Information and Automation*, pages 1194–1199, Aug 2015.

[51] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 957–964, May 2012.

[52] Kejian Wu, Ahmed Ahmed, Georgios A. Georgiou, and Stergios I. Roumeliotis. A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. 2015.

[53] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. 07 2014.

[54] Ji Zhang and Sanjiv Singh. Laser-visual-inertial odometry and mapping with high robustness and low drift. 08 2018.