

Mobile Robot Positioning via Visual and Inertial Sensor Fusion

by

Niraj Reginald

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2024

© Niraj Reginald 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Wail Gueaieb
Professor, School of Electrical Engineering and Computer Science,
University of Ottawa

Supervisor(s): Baris Fidan
Professor, Dept. of Mechanical and Mechatronics Engineering,
University of Waterloo
Ehsan Hashemi
Professor, Dept. of Mechanical Engineering,
University of Alberta

Internal Member: Soo Jeon
Professor, Dept. of Mechanical and Mechatronics Engineering,
University of Waterloo

Internal Member: Stewart McLachlin
Professor, Dept. of Mechanical and Mechatronics Engineering,
University of Waterloo

Internal-External Member: Yash Vardhan Pant
Professor, Dept. of Electrical and Computer Engineering,
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Niraj Reginald was the sole author for Chapters 1, 2, 3, 6 and 7 which were written under the supervision of Prof. Baris Fidan and Prof. Ehsan Hashemi and were not written for publication.

This thesis encompasses, in part, three manuscripts written for publication, as explained below. Exceptions to sole authorship of material are as follows:

Research presented in Chapter 4 has appeared in the following publication:

Publication 1: Reginald, N., Al-Buraiki O., Fidan B., Hashemi E., “Confidence estimator design for dynamic feature point removal in robot visual-inertial odometry,” in Proc. 48th Annual Conference of the Industrial Electronics Society, Brussels, Belgium, October 2022 (6 double column pages).

This research was conducted at the University of Waterloo by Niraj Reginald under the supervision of Prof. Baris Fidan and Prof. Ehsan Hashemi. Niraj Reginald developed the problem formulation, algorithm development, and real-world data based simulations, under supervision of Prof. Baris Fidan and Prof. Ehsan Hashemi. All co-authors (Prof. Baris Fidan, Prof. Ehsan Hashemi and Dr. Omar Al-Buraiki) participated in drafting the manuscript. Prof. Baris Fidan and Prof. Ehsan Hashemi provided intellectual inputs as subject matter experts.

Research presented in Chapter 5 has appeared in the following publications:

Publication 2: Reginald N., Al-Buraiki O., Fidan B., Hashemi E., “Augmented visual inertial wheel odometry through slip compensation,” Chapter 7 in Measurements and Instrumentation for Machine Vision, O. Sergiyenko and J.E. Miranda-Vega (eds.), CRC Press, pp. 157-170, 2024.

This research was conducted at the University of Waterloo by Niraj Reginald under the supervision of Prof. Baris Fidan and Prof. Ehsan Hashemi. Niraj Reginald developed the problem formulation, algorithm and real-world data based simulations, under close supervision of Prof. Baris Fidan and Prof. Ehsan Hashemi. All co-authors (Prof. Baris Fidan, Prof. Ehsan Hashemi and Dr. Omar Al-Buraiki) participated in drafting the manuscript. Prof. Baris Fidan and Prof. Ehsan Hashemi provided intellectual inputs as subject matter experts.

Publication 3: Reginald N., Al-Buraiki O., Choopojcharoen T., Fidan B., Hashemi E. “Visual Inertial Wheel Odometry With Slip Compensation and Dynamic Feature Elimination,” IEEE Sensors Journal, 2024, Submitted.

This research is a follow up of Publication 1 and was conducted at the University of Waterloo by Niraj Reginald under the supervision of Prof. Baris Fidan and Prof Ehsan Hashemi, and encompasses research material covered in Chapters 4 and 5. Niraj Reginald developed the problem formulation, theoretical derivation, algorithm, real-world data based simulations and indoor experimentation, under close supervision of Prof. Baris Fidan and Prof. Ehsan Hashemi. The co-authors Prof. Baris Fidan, Prof. Ehsan Hashemi and Mr. Thanacha Choopojcharoen participated in drafting the manuscript. Prof. Baris Fidan and Prof. Ehsan Hashemi provided intellectual inputs as subject matter experts.

Abstract

A fundamental prerequisite of mobile robots is the ability to accurately localize itself in a given environment. Accurate localization information is vital for a mobile robotic agent where different modules such as motion planning and control rely upon. Global Navigation Satellite systems (GNSS) are a popular mechanism to obtain geolocation of a robot in outdoor environments. However, GNSS systems can be unreliable in indoor/outdoor environments where GNSS signals struggle to penetrate such as urban canyons, indoor environments, tunnels and underground infrastructure etc. Therefore, localization by means of other sensory measurements and techniques is a requirement. The main purpose of this research is to develop an accurate robot localization system via multi-sensor fusion from available sensory information such as visual, inertial, and wheel encoder measurements.

As a solution to this requirement the fusion of monocular visual, inertial, and wheel encoder measurements has recently gained immense interest as a robot odometry and localization approach that overcomes the effects of navigation system uncertainties and variations. However, the wheel encoder measurements fused to the visual inertial wheel odometry (WO) system in this approach can be faulty mainly due to wheel slippages and other inherent errors. This thesis proposes a strategy for compensating wheel slip effects, based on a differential drive robot kinematics model. We use Gaussian process regression to learn the error between the WO model and the ground-truth for a set of training sequences. A deep kernel is constructed leveraging Long short term memory (LSTM) networks to capture the sequential correlations of odometry error residual. The learned WO error information is then used on the test sequences to correct the errors in WO. Then, the corrected WO measurements are utilized in a multi-state-constraint Kalman Filter based robot state estimation scheme. The enhancement is demonstrated via simulation experiments based on real-world data sets and indoor experimental evaluations using a test platform mobile robot.

In addition, the visual measurements are corrected via a feature point confidence estimator design to discard dynamic features in the feature matching process and subsequently for motion estimation. The development of the estimator design comprises of estimating the fundamental matrix using an inertial measurement unit to geometrically verify the matched confidence of visual key-points. Simulation results based on real world data sets confirm the improved accuracy of the overall designed localization scheme.

Acknowledgements

First and foremost, I'd like to thank my supervisors, Professor Baris Fidan and Professor Ehsan Hashemi, whose vast knowledge, professional guidance, and patience helped me build and finish my research projects. They were highly supportive and provided sincere, insightful suggestions that profoundly impacted my professional life.

Finally, I would like to thank my late father along with my mother for providing strength and support to overcome the challenges met.

Dedication

This is dedicated to my parents.

Table of Contents

Examining Committee Membership	ii
Author's Declaration	iii
Statement of Contributions	iv
Abstract	vi
Acknowledgements	vii
Dedication	viii
List of Figures	xiii
List of Tables	xvi
Nomenclature	xvii
1 Introduction	1
1.1 Problem Domain and Motivation	2
1.2 Objectives	3
1.3 Thesis Contributions	3
1.4 Thesis Outline	5

2	Background and Literature Review	6
2.1	Preliminaries of Localization	6
2.2	Visual Odometry	9
2.2.1	Feature Extraction and Tracking	10
2.2.2	Feature-based Visual SLAM	11
2.2.3	Direct Methods for Visual SLAM	12
2.3	Visual Inertial Navigation Systems	12
2.3.1	Inertial Kinematics	13
2.3.2	Visual-Inertial Sensor Fusion: Extended Kalman Filter Approach	15
2.3.3	Multi-state Constraint Kalman Filter	16
2.3.4	Batch Least Squares Estimation	18
2.3.5	Marginalization	19
2.3.6	Inertial Pre-integration	19
2.3.7	Incremental Optimization	20
2.3.8	Observability Properties of Visual Inertial Navigation Systems	21
2.3.9	State Initialization	22
2.3.10	Long-term Navigation Key Problems	23
2.4	Visual Inertial Wheel Odometry	24
3	Comparative Numerical Analysis of State of the Art VINS Algorithms	26
3.1	Simulation Setup	26
3.1.1	Metrics and Evaluation	27
3.2	Simulation Studies	27
3.3	Drawbacks of the Current State of the Art Methods	29
4	Dynamic Feature Point Elimination	33
4.1	The System and Problem Definition	34
4.1.1	Robot State Vector	35

4.1.2	Camera Measurements and Visual Odometry	35
4.1.3	Wheel Encoder Measurements	36
4.1.4	IMU Measurements	37
4.1.5	Scale Estimation	38
4.1.6	Gyroscope Bias Estimation	38
4.1.7	Accelerometer Bias Estimation	39
4.2	Overall Methodology	40
4.3	MSCKF Based Robot Odometry Estimation	42
4.4	IMU Based Confidence Estimation and Outlier Removal	43
4.5	Real-world Data Based Simulations on Dynamic Point Elimination	46
4.5.1	Dynamic Point Elimination	46
4.6	Summary and Conclusion	47
5	Integration of Wheel Slip Compensation to VINS	51
5.1	Problem and Approach	51
5.2	Wheel Slippage Estimation	52
5.2.1	Inertial Measurement Model	52
5.2.2	Wheel Encoder Measurements	54
5.3	Overall Methodology	55
5.4	Slip Compensation via Gaussian Process Regression	56
5.5	State Estimation	59
5.6	Real-World Data Based Simulation on Slip Compensation	59
5.6.1	Dataset Training	59
5.6.2	Results	61
5.7	Indoor Experimentation	63
5.8	Summary and Conclusion	65

6	Semantic Segmentation for Navigation	73
6.1	Introduction	73
6.2	Overall Methodology	74
6.3	Dynamic feature point elimination	77
6.4	Real-World Data Based Simulations	80
6.5	Summary and Conclusion	83
7	Conclusion and Future Work	86
7.1	Conclusions	86
7.2	Future Work and Directions	87
	References	89

List of Figures

2.1	Graph SLAM structure [45].	7
2.2	Pose graph example [45].	8
2.3	VO pipeline.	10
2.4	Overview of ORB-SLAM [102].	12
2.5	IMU frame of reference.	15
2.6	MSCKF outline.	17
2.7	Factor graph form of visual-inertial state estimation.	19
2.8	A typical VIWO system.	24
3.1	ORB-SLAM-3 on Euroc mh-01 trajectory top view.	29
3.2	ORB-SLAM-3 on Euroc mh-01 trajectory side view.	30
3.3	ORB-SLAM-3 on Euroc mh-01 trajectory position and scale drift.	31
3.4	ORB-SLAM-3 on Euroc dataset machine hall 01 sequence translation error and percentage.	32
3.5	ORB-SLAM-3 on Euroc dataset machine hall 01 sequence rotation error.	32
4.1	Proposed robot pose estimation method with dynamic feature point removal	41
4.2	MSCKF framework for robot pose estimation	43
4.3	Three view geometry	45
4.4	Comparison of robot trajectory 1 with and without dynamic point elimination for the cases of (a) VIO (b) VIWO	47

4.5	Comparison of robot trajectory 2 with and without dynamic point elimination for the cases of (a) VIO (b) VIWO.	48
4.6	Trajectory 1 translation and rotation error with 3-sigma error bounds.	49
4.7	Trajectory 2 translation and rotation error with 3-sigma error bounds.	49
5.1	Robot model.	55
5.2	Outline of the proposed VIWO scheme.	56
5.3	Slip compensation scheme.	57
5.4	Training loss propagation for each epoch illustrating model convergence.	61
5.5	Training time at each epoch to complete one cycle of training and optimization.	62
5.6	Slip compensation comparison	63
5.7	Slip compensation comparison for test sequence 2.	64
5.8	Trajectory error for test sequence 1.	65
5.9	Trajectory error for test sequence 2.	66
5.10	Trajectory plot for test sequence 1 with LSTM based Kernel. VIWO : Visual Inertial Wheel Odometry, DPE: Dynamic Point Elimination, WSC: Wheel Slip Compensation.	67
5.11	Trajectory plot for test sequence 2 with LSTM based Kernel. VIWO: Visual Inertial Wheel Odometry, DPE: Dynamic Point Elimination, WSC: Wheel Slip Compensation.	68
5.12	Trajectory error for test sequences with LSTM based deep Kernel.	68
5.13	Comparison of wheel speeds of the robot along with ground truth captured and trajectory estimated for a sample of training dataset. FL: Front Left, FR: Front Right, RL: Rear Left, RR: Rear Right.	69
5.14	Experimental setup, which includes the UGV and Optitrack Motion Capture system.	70
5.15	Experiment sequence 1 odometry trajectory comparison. DPE: Dynamic Point Elimination, SC: Slip Compensation (LSTM GPR)	70
5.16	Experiment sequence 2 odometry trajectory comparison. DPE: Dynamic Point Elimination, SC: Slip Compensation (LSTM GPR)	71
5.17	RMSE(m) for the experimental trajectories.	71

5.18	Experiment sequence 3 odometry trajectories comparison in gravel/sand based terrain.	72
6.1	Overall structure of dynamic object removal	75
6.2	Outline overview of Mask-RCNN network [50]. (a) In the first stage there are two networks. a backbone network (Resnet in this article) and a RPN (region proposal network). Region proposals are regions in the feature map which consists an object. (b) The second stage consists of the prediction of object class and bounding boxes	76
6.3	City scape dataset example segmentation.	77
6.4	Sample segmentation results on dynamic scenes from the test dataset . . .	78
6.5	Dynamic point removal, real world data example using semantic segmentation and geometric consistency.	80
6.6	Trajectory 1 localization estimation with and without Semantic Segmentation.	81
6.7	Trajectory 1 translation error plot in comparison to with and without Semantic Segmentation.	82
6.8	Trajectory 1 rotation error plot in comparison to with and without Semantic Segmentation.	83
6.9	Trajectory 2 localization estimation with and without Semantic Segmentation.	84
6.10	Trajectory 2 translation error plot in comparison to with and without Semantic Segmentation.	84
6.11	Trajectory 2 rotation error plot in comparison to with and without Semantic Segmentation.	85

List of Tables

2.1	IMU kinematic model parameters.	14
3.1	Simulation study EuRoc dataset.	28
3.2	Simulation results for ORB-SLAM-3 using TUM dynamic dataset.	30
4.1	Translation error with and without dynamic point elimination.	50
5.1	Sensor specifications for KAIST dataset.	60
5.2	Parameters for training GP	60
5.3	Jackal Sensor specifications.	65
6.1	Translation error comparison for with and without Semantic Segmentation implementation depicted in ARMSE (average root mean squared error).	82

Nomenclature

ATE absolute trajectory error [27](#), [28](#)

CNN Convolutional Neural Network [62](#), [73](#), [86](#), [87](#)

DNN Deep Neural Network [62](#)

DOF Degree of Freedom [13](#), [22](#)

EKF Extended Kalman Filter [1](#), [6](#), [15–17](#)

FOG Fiber Optics Gyro [55](#), [56](#)

GNSS Global navigation satellite system [12](#)

GP Gaussian Processes [66](#)

GPR Gaussin Process Regression [3](#), [4](#), [56](#), [65](#), [86](#), [87](#)

GPS Global Positioning System [1](#), [51](#)

IMU Inertial Measurement unit [xiii](#), [1–4](#), [9](#), [13–15](#), [18](#), [20](#), [23](#), [25](#), [29](#), [34](#), [37](#), [39](#), [48](#), [53](#), [55](#), [63](#), [77](#), [79](#), [81](#), [86](#)

INS Inertial Navigation System [2](#)

LSTM Long Short Term Memory [4](#), [62](#), [86](#), [87](#)

MAP Maximum-a-Posteriori [22](#)

MSCKF Multi-state constraint Kalman filter [xiii](#), [4](#), [13](#), [16](#), [17](#), [34](#), [47](#), [51](#), [58](#), [59](#), [61](#), [65](#), [75](#)

NEES Normalized estimation error squared [27](#)

RANSAC Random Sample Consensus [4](#), [29](#), [33](#), [34](#), [44](#), [46](#), [48](#), [78](#), [83](#), [86](#), [87](#)

RMSE Root mean squared error [27](#), [46](#), [64](#), [81](#)

RPE Relative pose error [27](#)

SFM Structure from Motion [5](#), [38](#)

SLAM Simultaneous Localization and Mapping [1](#), [2](#), [6](#), [7](#), [9–11](#), [26](#), [30](#), [74](#)

SSD Sum of squared differences [11](#)

VINS Visual Inertial Navigation System [2](#), [3](#), [5](#), [13](#), [18](#), [19](#), [21–24](#), [26](#), [27](#), [29](#), [33](#), [51](#), [52](#)

VIO Visual inertial odometry [2](#), [3](#), [18](#), [19](#), [24–27](#), [38](#), [55](#)

VISLAM Visual inertial simultaneous localization and mapping [2](#), [28–30](#), [51](#)

VIWO Visual inertial wheel odometry algorithms [3](#), [4](#), [51](#), [55](#), [56](#), [65](#)

VO Visual Odometry [xiii](#), [2–4](#), [9](#), [10](#), [25](#), [27](#), [55](#), [83](#), [86](#)

VSLAM Visual Simultaneous Localization and Mapping [2](#), [10](#), [11](#), [24](#), [26](#), [28](#), [30](#), [33](#)

WO Wheel odometry [2](#), [4](#), [5](#), [9](#), [24](#), [25](#), [34](#), [37](#), [39](#), [40](#), [42](#), [46–48](#), [51](#), [52](#), [55](#), [56](#), [59](#), [65](#), [86](#), [87](#)

Chapter 1

Introduction

Mobile robots are becoming increasingly popular and necessary because of the various applications they have to offer. Active and emerging applications are used in various operations, including medical, warehousing, cleaning, navigation, autonomous driving, and agriculture. This is primarily due to their ability to accurately estimate their position in an unknown environment while simultaneously navigating and creating a map of the region. Without location information, path planning cannot determine where the robot is heading and its current location with respect to the destination. The aforementioned capabilities are mainly due to the advances in [SLAM](#) (Simultaneous Localization and Mapping) techniques. Hence, [SLAM](#) has been a very popular research area. During its initial phase between 1986 to 2004, probabilistic approaches such as [EKF](#) (Extended Kalman Filter), Particle Filter, and Maximum likelihood estimation were used. Then from year 2004 to 2015 came the "algorithmic-analysis" age where researchers mostly studied the fundamental properties of [SLAM](#) such as consistency, observability, and convergence. It was also during this period that the researchers sustained the idea of sparsity for the efficient computation of [SLAM](#) solvers [14]. The current existing state-of-the-art [SLAM](#) methods can handle thousands of landmarks at a given time [37]. Another factor that contributed to this effort was the improvement in hardware, processing power, sensors, etc.

One of the key outputs of a SLAM algorithm is the estimation of a robot's pose as a pose vector, which is the combination of the location and orientation of the robot. To carry out this operation many sensors have been discussed in literature. During the early stages of [SLAM](#) sensors such as [IMU](#) (Inertial Measurement Unit), [GPS](#) (Global Positioning System), Rotary Encoders, Laser Range sensors, and cameras were used independently or in combination. Between 2010 and 2016, [SLAM](#) algorithms consisting of only cameras were

popular due to the simple sensor configuration. Also, the technical difficulties associated with it were lower compared to other types of sensors [132].

Algorithms of this nature are called **VSLAM** (Visual Simultaneous Localization and Mapping) algorithms. The odometry obtained from these algorithms is called **VO** (Visual Odometry), which is the determination of the ego-motion of an object using a single or multiple set of cameras [123]. In the recent past, the trend has been to fuse **IMU** with a camera to obtain better results, where the camera and **IMU** supplement each other to compensate for each other’s disadvantages.

1.1 Problem Domain and Motivation

Using a camera as a sensor for **SLAM** problems has many advantages. Cameras in general consume less power, are lightweight, cost comparatively less, and provide very rich information. However, the camera does suffer from errors due to unavoidable circumstances such as illumination changes, distortion due to fast movements, etc. [3]. Inertial sensors are known to provide good signals at high frequency during fast movements. However, they are prone to drift due to errors accumulated through the double integration process to acquire position information. On the other hand, cameras provide an accurate estimated ego-motion in low frequency for long-term steady movements but suffer from the aforementioned scenarios. Hence, the objective of using visual-inertial sensor fusion is to overcome the fundamental limitations of **VSLAM** and **INS** (Inertial Navigation System). Accordingly, navigating using a combination of camera and **IMU** is called a **VINS** (Visual-Inertial Navigation System) system.

One of the key problems associated with **VSLAM** and **VISLAM** (Visual inertial simultaneous localization and mapping) is that they use static features in the environment, which can lead to errors in dynamic/semi-static environments. Therefore, it is important to devise a methodology to eliminate dynamic objects from the scene, so that **VINS** becomes flexible and robust to operate in a heavily dynamic environment.

Another inherited problem is that **VSLAM** and **VISLAM** algorithms require a sufficient amount of lighting to detect visual features. In indoor environments, there can be a drastic contrast in illumination and lighting conditions, making the extraction of meaningful features from the environment a challenging task. Especially in indoor environments, there can be instances where the environment has limited features. Such an environment with insufficient features would not allow the robot to localize properly. Hence, fusing **VIO** (Visual-Inertial Odometry) with additional sensory information such as **WO** (Wheel Odometry) has the potential to improve the accuracy of the overall localization.

Furthermore, sensor fusion algorithms such as [VINS](#) are very sensitive to time synchronization, a point that the developer must be concerned about to ensure robust and accurate estimation [139]. In addition, a robust and fast initialization mechanism is required for better overall accuracy and robustness.

1.2 Objectives

[VIO](#) is a well-studied problem in current literature. As an emerging topic to improve accuracy and overcome observability issues of [VIO](#) using [VIWO](#) (Visual Inertial Wheel odometry) is proposed. However, most of the current [VIO](#) and [VIWO](#) algorithms are wrapped around key assumptions. Firstly, in the [VO](#) front end, the scene is assumed to be static. Hence, a challenge [VIO/VIWO](#) algorithms face is navigating through environments that have highly dynamic objects. Since the visual front-end of these algorithms relies on feature matching of static objects, the dynamic objects will increase the error of motion estimation of [VO](#). However, [IMU](#) and other sensors attached to a mobile robot do not rely on feature matching for motion estimation. Hence, it is possible to use other available sensory information as a confidence estimator to detect and eliminate dynamic feature points for motion estimation.

In addition, for [VIWO](#) algorithms, it is assumed ideal rolling scenarios for the wheels. However, slippage inherently exists and can adversely affect localization accuracy. Hence, by compensating for slippage the overall accuracy of [VIWO](#) can be improved.

Considering these challenges, the main objective of this thesis is to develop accurate localization schemes achieving the following sub-objectives.

1. Development of a feature point confidence estimator to the [VO](#) front end to eliminate dynamic feature points in the feature matching process for motion estimation
2. Development of a data-driven slip compensation scheme using deep kernel based [GPR](#)

1.3 Thesis Contributions

The following research articles have appeared for publication and comprise the technical content of this thesis:

1. Reginald, N., Al-Buraiki O., Fidan B., Hashemi E., “Confidence estimator design for dynamic feature point removal in robot visual-inertial odometry,” in Proc. 48th Annual Conference of the Industrial Electronics Society, Brussels, Belgium, October 2022 (6 double column pages).
2. Reginald N., Al-Buraiki O., Fidan B., Hashemi E., “Augmented visual inertial wheel odometry through slip compensation,” Chapter 7 in Measurements and Instrumentation for Machine Vision, O. Sergiyenko and J.E. Miranda-Vega (eds.), CRC Press, pp. 157-170, 2024.
3. Reginald N., Al-Buraiki O., Choopojcharoen T., Fidan B., Hashemi E. “Visual Inertial Wheel Odometry With Slip Compensation and Dynamic Feature Elimination,” IEEE Sensors Journal, 2024, Submitted.

Specifically, the thesis contributions are of twofold:

1. Development of a module for checking the confidence of feature matching:

This development encompasses using the **IMU** motion model to check the geometric consistency of feature points. The **IMU** unit translation and rotation is used to estimate the geometric Fundamental matrix to check the visual feature point’s geometric consistency to classify them as static or dynamic. Once the dynamic feature points are eliminated, the refined feature points are processed through the **VO** front end to estimate the camera motion for improved accuracy. The improved **VO** is used for robot pose correction via a multi-state constraint Kalman filter (**MSCKF**). If the IMU data is deemed invalid as a redundant method **RANSAC** (Random Sample Consensus) is used to eliminate outliers based on trifocal geometry.

2. Development of a data-driven slip compensation scheme using deep kernel based **GPR** (Gaussian Process Regression):

We estimate the **WO** residual at each sampling time given the velocity and heading angle, assuming planar motion using a **GPR** approach. The kernel is extended using **LSTM** (Long Short Term Memory) layers to better capture the sequential correlations of odometry. Based on this slip compensated **WO** the **MSCKF** is updated to obtain **VIWO** estimate.

1.4 Thesis Outline

Background information and references to current existing literature have been discussed in Chapter 2. These can be considered as the main tools required for implementing [VINS](#) algorithms. Chapter 3 includes a comparison between the current state-of-the-art methods in [VINS](#) via experimental evaluation using public datasets. Chapter 4 includes a methodology for a confidence estimator design to remove dynamic point data association for visual [SFM](#) (Structure From Motion) and thereby improve the overall localization accuracy. The enhancement to odometry using the proposed method is validated using real-world data simulation studies. Chapter 5 includes augmenting slip-compensated [WO](#) measurements to improve localization accuracy. Wheel slippage error is estimated using a data-driven Gaussian Process Regression approach. The enhancement to the odometry is demonstrated via real-world data simulations and indoor experimental studies. In addition, Chapter 6 provides the design of an additional filter utilizing semantic segmentation to eliminate dynamic feature points. Finally, Chapter 7 provides the concluding remarks and encapsulates the future research directions.

Chapter 2

Background and Literature Review

2.1 Preliminaries of Localization

The main goal of [SLAM](#) is to construct a map with information from sensors while simultaneously localizing itself to the map. In terminology, Full SLAM means the posterior estimation of the entire path of the robot whereas Online SLAM refers to the estimation of the current posterior only [\[134\]](#).

In general, there are three main principle paradigms to tackle the problems associated with [SLAM](#). They are respectively [EKF](#) based methods, particle filter-based methods, and graph-based optimization methods. Of these paradigms, [EKF](#) based [SLAM](#) methods can be computationally ineffective due to scaling limitations. Furthermore, the covariance matrix is quadratic to the number of features. In comparison, particle filter-based methods try to mitigate this issue relating to the feature correlations in the map. The individual landmarks in the map are independent and hence become uncorrelated. FastSLAM [\[98\]](#) is one of the methods that use particle filter to tackle the [SLAM](#) problem. However, the number of particles can grow infinitely, especially for robotic agents who are seeking to map multiple nested loops of the environment. The graph-based [SLAM](#) approach is currently the most popular method amongst researchers, as it tries to address the full problem of [SLAM](#) through highly efficient sparse nonlinear optimization [\[130\]](#).

The [SLAM](#) problem can be illustrated using a graph where every node in the graph corresponds to a pose of the robot during mapping, and an edge between two nodes corresponds to a spatial constraint between them. Here, the final objective is to build the graph and find a node configuration that minimizes the error introduced by the constraints.

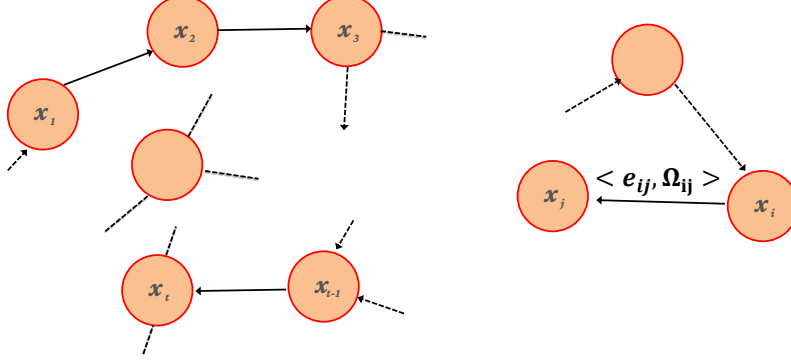


Figure 2.1: Graph SLAM structure [45].

Figure 2.2 illustrates one single edge of the graph. Here, two arbitrary nodes consisting of robot poses of the graph are denoted as $\mathbf{x}_i$ and $\mathbf{x}_j$. For simplicity purposes, consider the robot poses $\in \mathbb{R}^2 \times \mathbb{S}^1$ in 2D. If we observe the node $\mathbf{x}_j$ from $\mathbf{x}_i$, our observation model denotes that node $\mathbf{x}_j$ is in the location described by a mean representing an observation $\mathbf{z}_{ij}$ and covariance information matrix $\mathbf{\Omega}_{ij}$. Hence, in the context of a SLAM problem, the information matrix will consist of many edges $\langle \mathbf{e}_{ij}, \mathbf{\Omega}_{ij} \rangle$, where $\mathbf{e}_{ij} = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}$ and $\hat{\mathbf{z}}_{ij}$ is the predicted observation. Let $\mathbb{C}$ be the set of pairs of indices for which a constraint (observation) $\mathbf{z}$ exists. Then, the graph SLAM approaches targets to find the nodes that minimize the negative log-likelihood $F(\mathbf{x})$ of all observations

$$F(\mathbf{x}) = \sum_{i,j \in \mathbb{C}} F_{ij}(\mathbf{x}) = \sum_{i,j \in \mathbb{C}} \mathbf{e}_{ij}(\mathbf{x})^\top \mathbf{\Omega}_{ij} \mathbf{e}_{ij}(\mathbf{x}), \quad (2.1)$$

thus, it looks to solve the following equation

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} F(\mathbf{x}). \quad (2.2)$$

$\mathbf{e}_{ij}(\mathbf{x})$ is the error as the function of the whole state vector $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ where $\mathbf{x}_i \in \mathbb{R}^2 \times \mathbb{S}^1$ consist of robot poses in 2D.

To further analyze the meaning of each error, a map between each pose to its corresponding homogeneous transformation matrix $\mathbf{H}$ can be denoted by

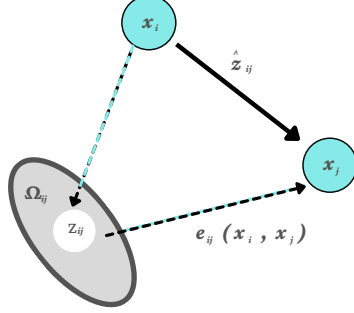


Figure 2.2: Pose graph example [45].

$$\mathbf{T}(\cdot) : \mathbb{R} \times \mathbb{S}^1 \rightarrow \text{SE}_2 \quad (2.3)$$

An error term $\mathbf{e}_{ij}$ will be zero if $\mathbf{T}(\mathbf{z}_{ij}) = \mathbf{T}(\mathbf{x}_i)^{-1}\mathbf{T}(\mathbf{x}_j)$ where the observation is the same as relative graph configuration. Here, $\mathbf{H}(\mathbf{x}_i)$ and $\mathbf{T}(\mathbf{x}_j)$ are poses of the robot represented as homogeneous transformation matrices. A least square approach can be used to solve the minimization problem 2.2. The error function of the state vector consisting of poses is linearized as

$$\mathbf{e}_{ij}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{e}_{ij}(\mathbf{x}) + \mathbf{J}_{ij}\Delta\mathbf{x}, \quad (2.4)$$

where

$$\mathbf{J}_{ij} = \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}} = [\mathbf{0} \quad \dots \quad \mathbf{A}_{ij} \quad \dots \quad \mathbf{B}_{ij} \quad \dots \quad \mathbf{0}]$$

$$\mathbf{A}_{ij} = \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_i}$$

$$\mathbf{B}_{ij} = \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_j}$$

$$\Delta\mathbf{x} = \mathbf{x}^* - \check{\mathbf{x}}$$

,and $\check{\mathbf{x}}$ is an initial guess of the robot poses.

Since $\mathbf{e}_{ij}(\mathbf{x})$ only depend on $\mathbf{x}_i$ and, $\mathbf{x}_j$ the jacobian function $\mathbf{J}_{ij}$ will have a sparse nature, which has a great effect on computational capacity. From this, the sparse matrices $\mathbf{b}$ and $\mathbf{H}$ is derived to obtain a linear system of equations as

$$\mathbf{H} = \sum_{ij} \mathbf{H}_{ij} = \sum_{ij} \mathbf{J}_{ij}^{ij} \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}, \quad (2.5)$$

where the coefficient matrix of an edge is

$$\mathbf{H}_{ij} = \begin{bmatrix} \vdots \\ \mathbf{A}_{ij}^\top \\ \vdots \\ \mathbf{B}_{ij}^\top \\ \vdots \end{bmatrix} \boldsymbol{\Omega}_{ij} \begin{bmatrix} \cdots & \mathbf{A}_{ij} & \cdots & \mathbf{B}_{ij} & \cdots \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} & \mathbf{A}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} \\ \mathbf{B}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} & \mathbf{B}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} \end{bmatrix}. \quad (2.6)$$

Below, Algorithm 1 summarizes the outline of graph based SLAM illustrated above. Here, the convergence condition is fulfilled once the system of equations is solved for $\Delta \mathbf{x}$ to update $\mathbf{x}$. This is usually solved using techniques such as sparse Cholesky factorization.

Algorithm 1: Optimize($\mathbf{x}$)

Result: optimized $\mathbf{x}$
while *not converged* **do**
 $(\mathbf{H}, \mathbf{b}) \leftarrow \text{buildLinearSystem}(\mathbf{x})$
 $\Delta \mathbf{x} \leftarrow \text{SolveSparse}(\mathbf{H} \Delta \mathbf{x} = -\mathbf{b})$
 $\mathbf{x} \leftarrow \mathbf{x} + \Delta \mathbf{x}$
end

2.2 Visual Odometry

Visual odometry (VO) for any agent can be defined as estimating its ego-motion using monocular or multiple cameras attached to the rigid body of the agent. VO incrementally estimates the robot agent's position by looking at the changes of image frames it perceives, which are induced by its motion. Thereby, this is analogous to wheel odometry (WO) where the motion of the robot agent is incrementally estimated by integrating the number of rotations of its wheels. Compared to WO, the advantage of VO is that it is not affected by wheel slippages [123]. Figure 2.3 outlines the main components of VO. Chapter 4 demonstrates the importance of feature matching for VO front end by developing a mechanism to eliminate dynamic feature points by checking their geometric consistency using IMU motion model.

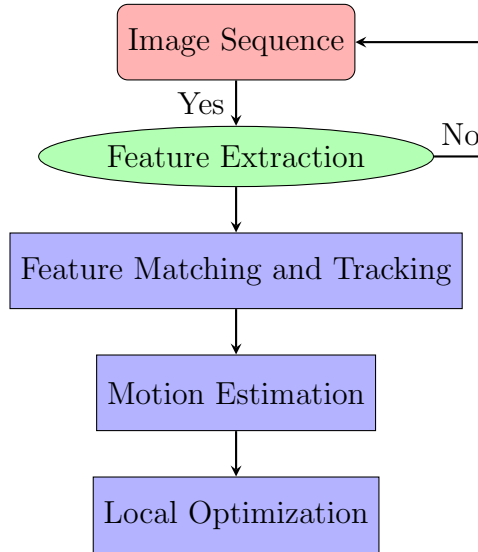


Figure 2.3: VO pipeline.

VO is accomplished in real-time by using a sequence of ordered images to estimate the motion of the camera. On top of estimating the motion of, the camera VSLAM also does the mapping and recognizes the places that have already been visited.

Direct SLAM methods do not search for key points in images, but instead look for image intensities to estimate the location of the surroundings. On the other hand, feature-based methods use feature extraction methods and depend on the features to estimate the location of the surroundings.

2.2.1 Feature Extraction and Tracking

During the feature detection stage of VO and VSLAM pipeline, the images are searched for key points that are likely to match well in other images. In Literature, there are many corner detectors such as Moravec, Forstner, Harris, Shi-Tomasi, FAST etc; and blob detectors such as SIFT, SURF, and Censur etc.[40]. A key point is a locally distinct location in an image. The feature descriptor summarizes the local structure around the key point (usually as a vector). In other words, they find locally distinct points given an image. Corners are considered highly distinct points. This is due to the high image gradients that corners have. They are invariant to translation, rotation, and illumination changes. Corners can be defined as edges in roughly orthogonal directions. An edge can

be interpreted as a sudden brightness change. Corners are found by searching for intensity changes in two directions. By computing the sum of squared differences (SSD) around the neighbors (x, y) image coordinates, the key interest points in the image are obtained. The function for computing SSD is given as

$$f(x, y) = \sum_{(u, v) \in W_{x, y}} \left\{ I \left(\begin{bmatrix} u \\ v \end{bmatrix} \right) - I \left(\begin{bmatrix} u + \delta u \\ v + \delta v \end{bmatrix} \right) \right\}^2, \quad (2.7)$$

where u, v are image coordinates. $\delta u, \delta v$ are the shift in x and y direction of the window W . $I(\cdot)$ is the function that calculates the intensity value of a given image coordinate. The areas with maximum SSD values can be thought of as areas that have higher gradients.

Unlike finding key interest points in images, the feature descriptors are used to describe a point of interest uniquely. As an example, SIFT Key-points are generated using the difference of Gaussian's approach over scale-space pyramid. SIFT stands for scale-invariant feature transform where over different image pyramid levels Gaussian smoothing is applied [89]. SIFT tries to find scale-invariant feature points from the local maxima of scale space.

2.2.2 Feature-based Visual SLAM

Feature-based SLAM uses the corner or blob detectors and tries to find their correspondences of them in subsequent images. Since only certain key points or features of the image are taken into account, the feature-based methods can be identified as sparse methods. Due to their sparse nature, feature-based methods are widely used in literature compared to direct methods. These algorithms use the found key points or features to obtain 3D pose estimates through geometric error. Given a 3D point of a feature $\mathbf{p}_i$ and its corresponding 2D point $\bar{\mathbf{p}}_i$, feature-based VSLAM aims to minimize the image re-projection error $\mathbf{e}_r$ as

$$\mathbf{e}_r = \bar{\mathbf{p}}_i - \boldsymbol{\pi}_m(\mathbf{C}\mathbf{p}_i + \mathbf{t}), \quad (2.8)$$

where $\boldsymbol{\pi}_m$ is a function for projecting world 3D point coordinates into 2D camera coordinates. $\mathbf{C}, \mathbf{t}$ are the rotation matrix and translation vector respectively.

One of the most popular approaches in feature-based SLAM is ORB-SLAM [102]. The front-end process uses the sensor data and constructs the graph by using nodes and edges. ORB-SLAM also recognizes if the current scene was repeated before to perform loop closure. The back-end process does graph optimization using this loop information. Figure 2.4 summarizes ORB-SLAM in a holistic viewpoint. Some other examples of feature-based VSLAM algorithms are MonoSLAM [29] and S-PTAM [110].

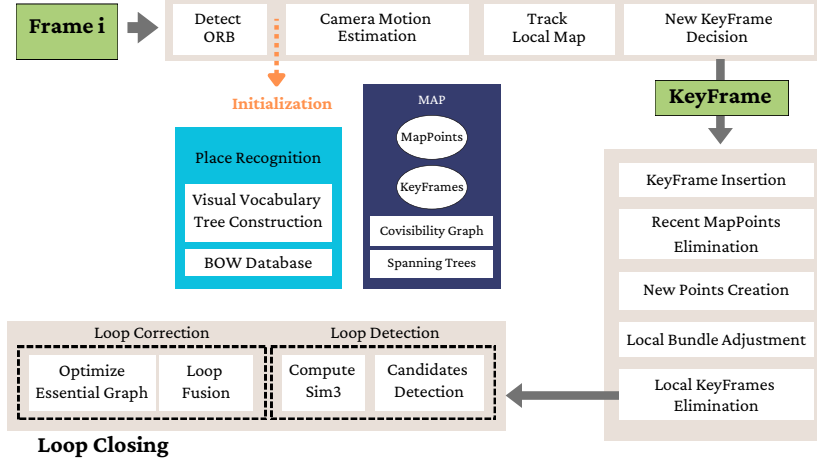


Figure 2.4: Overview of ORB-SLAM [102].

2.2.3 Direct Methods for Visual SLAM

The direct methods eliminate the pre-processing steps mentioned in the feature-based methods such as feature extraction. Direct methods utilize the displacement of every pixel intensity from a part or full image to estimate the pose by minimizing photo-metric error e_p as

$$e_p = I_i(\bar{p}) - I_j(\pi_d(\bar{p}, D(\bar{p}), \mathbf{T}_{ij})), \quad (2.9)$$

where $I_j(\pi_d)$ is a function which projects a 2D point from an image with its corresponding inverse depth value $D(\bar{p})$, into a new image by the transformation of the camera $\mathbf{T}_{ij}$.

Some examples of these direct methods are DTAM [105] and LSD-SLAM [37]. These methods have an advantage over feature-based methods in low-textured environments, as they do not depend on sparse features.

2.3 Visual Inertial Navigation Systems

In recent years, there has been great interest in multi-sensor fusion-based algorithms for localization problems in the field of mobile robotics. This is especially important in the case of global navigation satellite system (GNSS) as it is unavailable in indoor scenarios.

VINS combine IMU and camera measurements in real-time to output 6-DOF position and orientation information along with a 3D map of the environment. In literature, to tackle the VINS problem there exist multiple methods ranging from filtering-based methods to optimization-based methods as well as deep learning-based methods [24, 126]. A common point that these state-of-the-art methods proposed in recent times is that they follow a tightly coupled sensor fusion scheme, which provides better accuracy compared to loosely coupled methods.

One of the well-known methods for sensor fusion of VINS systems is the MSCKF (multi-state constraint Kalman filter) technique [101]. An advantage of visual sensing is that it can provide highly rich information, whereas feature extraction methods can provide information of hundreds of features. If these features are utilized properly, one can obtain good localization results. However, the problem here is that the high volume of data leads to high computational demands, in which real-time implementation can be challenging. In contrast to other methods, MSCKF does not need 3D feature-based points in the filter state vector, making them computationally efficient and suitable for real-time implementation. Enhanced MSCKF based algorithms have gained attention in recent times for this matter [125].

As mentioned in the introduction, another drawback of low-level feature-based methods is that sometimes they lack robustness when exposed to textureless areas. Due to this reason optimization-based techniques are utilized as an alternative approach to compensate for this issue [65]. A drawback of such nonlinear optimization methods is that they require solving a non-linear least square optimization problem which can be highly computationally expensive.

2.3.1 Inertial Kinematics

The inertial kinematics of an IMU are briefly discussed below. Figure 2.5 shows the IMU frame of reference $\{I\}$ along with the global frame of reference $\{G\}$. The parameter notation descriptions for the below equations are provided in table 2.1. The IMU measurement models containing the measured rotational velocity ω_m and measured linear acceleration $\mathbf{a}_m$ are given by [23]

$$\omega_m = {}^I\omega_I^G + \mathbf{b}_g + \mathbf{n}_g, \quad (2.10)$$

$$\mathbf{a}_m = \mathbf{C}({}^I\mathbf{q}_G)({}^G\mathbf{a}_I^G - {}^G\mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a, \quad (2.11)$$

where ${}^I\boldsymbol{\omega}_I^G = [\omega_x \ \omega_y \ \omega_z]^\top$ is the IMU's angular velocity relative to frame $\{G\}$ and expressed in frame $\{I\}$, ${}^G\mathbf{a}_I^G$ is the linear acceleration of the IMU relative to frame $\{G\}$ and with respect to frame $\{G\}$.

Notation	Representation
$\mathbf{q}$	orientation in form of quaternion
$\mathbf{C}$	orientation in form of rotation matrix
$\mathbf{p}$	position
$\mathbf{v}$	linear velocity
$\mathbf{a}$	linear acceleration
$\boldsymbol{\omega}$	angular velocity
$\mathbf{b}_a$	accelerometer bias
$\mathbf{b}_g$	gyroscope bias
$\mathbf{g}$	gravitational acceleration
$\mathbf{n}_a$	accelerometer measurement noise
$\mathbf{n}_g$	gyroscope measurement noise
$\mathbf{n}_{bg}$	gyroscope bias process noise
$\mathbf{n}_{ba}$	accelerometer bias process noise

Table 2.1: IMU kinematic model parameters.

Furthermore, from this IMU measurement model we denote the states of the IMU given in the continuous-time state equations given as

$${}^I\dot{\mathbf{q}}_G = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_m - \mathbf{b}_g - \mathbf{n}_g){}^I\mathbf{q}_G, \quad (2.12)$$

$${}^G\dot{\mathbf{p}}_I^G = {}^G\mathbf{v}_I^G, \quad (2.13)$$

$${}^G\dot{\mathbf{v}}_I^G = \mathbf{C}({}^I\mathbf{q}_G)^\top (\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) + {}^G\mathbf{g}, \quad (2.14)$$

$$\dot{\mathbf{b}}_g = \mathbf{n}_{bg}, \quad (2.15)$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{ba}, \quad (2.16)$$

where $\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\boldsymbol{\omega}_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^\top & 0 \end{bmatrix}$, $\boldsymbol{\omega}_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$.

Using these continuous time state equations, the state evolution model is written in

simplified form as [80]

$$\dot{\mathbf{x}} = \mathbf{f}_c(\mathbf{x}, \mathbf{u} - \mathbf{n}), \quad (2.17)$$

$$\mathbf{u} = [\boldsymbol{\omega}_m^\top \quad \mathbf{a}_m^\top]^\top, \quad (2.18)$$

where $\mathbf{x} = [\mathbf{q}_G^\top \quad (\mathbf{p}_I^G)^\top \quad (\mathbf{v}_I^G)^\top \quad \mathbf{b}_g^\top \quad \mathbf{b}_a^\top]^\top$. Here, $\mathbf{q}_G$ is the unit quaternion describing the rotation from frame $\{G\}$ to frame $\{I\}$. $\mathbf{v}_I^G, \mathbf{p}_I^G$ are the IMU velocity and position relative to and with respect to frame $\{G\}$. The discrete-time version of this equation is denoted as

$$\mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1:k}) + \mathbf{n}_k. \quad (2.19)$$

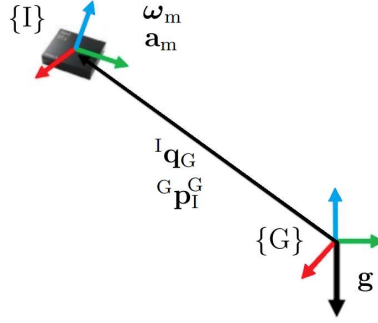


Figure 2.5: IMU frame of reference.

2.3.2 Visual-Inertial Sensor Fusion: Extended Kalman Filter Approach

Extended Kalman filter (EKF) is one of the popular methods used for state estimation tasks. It is a two-step recursive algorithm that intakes the incoming measurements and estimates an unknown state $\mathbf{x}$. The propagation step of the EKF algorithm is written as

$$\mathbf{u}_{k-1} = [\boldsymbol{\omega}_{m,k-1}^\top \quad \mathbf{a}_{m,k-1}^\top]^\top, \quad (2.20)$$

$$\mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}), \quad (2.21)$$

where $\mathbf{x}_{k-1} \sim \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1})$ is the Gaussian prior distribution and $\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^-, \mathbf{P}_k^-)$ is the propagated distribution. Here $(\hat{\cdot})$ denotes that it is an estimate. The function $\mathbf{f}_d$ which takes the previous state $\mathbf{x}_{k-1}$, the inertial measurements $\mathbf{u}_{k-1}$ and measurement noise $\mathbf{w}_{k-1}$ and brings the state forward from $k-1$ to k . Next, the update state of the [EKF](#) algorithm is shown below where the camera measurements denoted by $\mathbf{z}_m$ are given as

$$\mathbf{z}_m = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}_k, \quad (2.22)$$

is used to correct the state. As a result, we end up with a corrected state and co-variance $\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k^+, \mathbf{P}_k^+)$. Here the nonlinear measurement function $\mathbf{z}_m$ is linearized as

$$\mathbf{z}_m = \mathbf{h}(\hat{\mathbf{x}}_k^-) + \mathbf{H}_k \tilde{\mathbf{x}} + \mathbf{n}_k, \quad (2.23)$$

$$\mathbf{r}_k = \mathbf{H}_k \tilde{\mathbf{x}} + \mathbf{n}_k, \quad (2.24)$$

$$\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k). \quad (2.25)$$

This is to be used in the [EKF](#) algorithm where $\mathbf{H}_k$ denotes the jacobian function and $\tilde{\mathbf{x}}$ denotes the error between the true and current estimate of $\mathbf{x}$. Hence this linearized measurement is used in the standard [EKF](#) update as

$$\text{state: } \hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \mathbf{r}, \quad (2.26)$$

$$\text{co-variance: } \mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-, \quad (2.27)$$

where

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top + \mathbf{R}_k$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\top \mathbf{S}_k^{-1}$$

$$\mathbf{r}_k = \mathbf{z}_m - \mathbf{h}(\hat{\mathbf{x}}_k^-)$$

2.3.3 Multi-state Constraint Kalman Filter

Unlike the [EKF](#), the multi-state constrained Kalman filter ([MSCKF](#)) [101] is capable of updating features without inserting their estimates into the state vector. This is a major advantage as it reduces the complexity of the algorithm and increases computational efficiency. The brief outline of the [MSCKF](#) algorithm is shown in Figure (2.6). [MSCKF](#) is

a two-step process in which in the first step it propagates the state forward in time using the inertial information and in the second step it is corrected using the visual information. Given a set of measurements m from the camera measurement functions $\mathbf{z}_1 \dots \mathbf{z}_m$ as

$$\mathbf{z}_1 = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f_1}) + \mathbf{n}_{1,k}, \quad (2.28)$$

$\vdots$

$$\mathbf{z}_m = \mathbf{h}(\mathbf{x}_k, {}^G\mathbf{p}_{f_m}) + \mathbf{n}_{m,k}. \quad (2.29)$$

They are linearized and stacked together as

$$\tilde{\mathbf{z}}_{1:m} = \mathbf{H}_x \tilde{\mathbf{x}}_k + \mathbf{H}_f {}^G\tilde{\mathbf{p}}_f + \mathbf{n}_k, \quad (2.30)$$

where ${}^G\mathbf{p}_{f_i}$ is the 3D feature position in frame $\{G\}$, $\mathbf{n}$ is the unknown pixel noise.

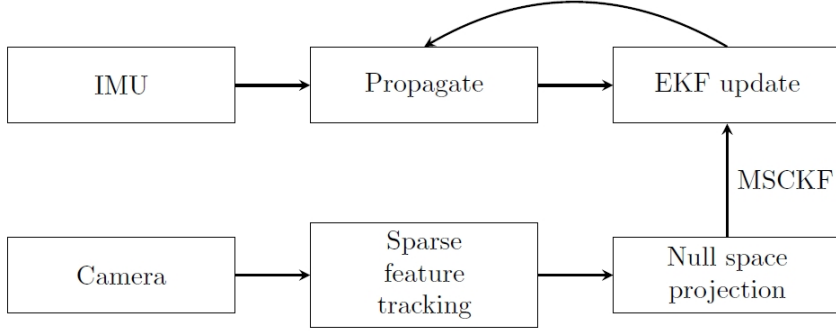


Figure 2.6: MSCKF outline.

By computing the null-space projection of the jacobian function $\mathbf{H}_f$ and defining as $\mathbf{Q}_n$ we state Eq.(2.30) as

$$\mathbf{Q}_n^\top \tilde{\mathbf{z}}_{1:m} = \mathbf{Q}_n^\top \mathbf{H}_x \tilde{\mathbf{x}}_k + \mathbf{Q}_n^\top \mathbf{H}_f {}^G\tilde{\mathbf{p}}_f + \mathbf{Q}_n^\top \mathbf{n}_k. \quad (2.31)$$

By doing so, the dependency of features in Eq.(2.30) is removed and simplified as

$$\mathbf{Q}_n^\top \tilde{\mathbf{z}}_{1:m} = \mathbf{Q}_n^\top \mathbf{H}_x \tilde{\mathbf{x}}_k + \mathbf{Q}_n^\top \mathbf{n}_k. \quad (2.32)$$

The nullspace projection causes the jacobian function $\mathbf{H}_f \rightarrow 0$. Thus, this creates a function that only depends on measurements. After computing this new measurement, the standard EKF update is performed to correct the state.

2.3.4 Batch Least Squares Estimation

Instead of looking at the [VIO](#) problem from one time-step to the next time-step, we can approach it in a factor graph perspective, as shown in Fig.(2.7). In this factor graph perspective, each node describes the state $\mathbf{x}_i$ at each time step along with the 3D position of the feature ${}^G\mathbf{p}_f$ with respect to the global coordinate frame $\{G\}$. The edges describe how each state relates to each other. $\mathbf{z}_i$ denotes the visual measurement and $\mathbf{u}_i$ denotes the inertial pre-integration. Here, the prior edge $\mathbf{P}_0$ denotes the initial covariance matrix of the factor graph and anchors it. Below are three different cost functions needed to solve for visual-inertial state estimation. The prior cost J_{prior} constraints the initial state $\mathbf{x}_0$. The inertial cost noted as J_{IMU} is the cost to evolve the previous [IMU](#) state at $k-1$ to the current state k . The cost function J_{camera} denotes the cost arising from the camera feature projection model.

$$J_{\text{prior}} = (\mathbf{x}_0 - \mathbf{x}'_0)^\top \mathbf{P}_0^{-1} (\mathbf{x}_0 - \mathbf{x}'_0) \quad (2.33)$$

where $\mathbf{x}_0 = \mathbf{x}'_0 + \mathbf{n}_k$

$$J_{\text{IMU}} = (\mathbf{x}_k - \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1}))^\top \mathbf{Q}_k^{-1} (\mathbf{x}_k - \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1})) \quad (2.34)$$

where $\mathbf{f}_d$ is the [IMU](#) state propagation function

$$J_{\text{camera}} = (\mathbf{z}_m - \mathbf{h}(\mathbf{x}_k))^\top \mathbf{R}_k^{-1} (\mathbf{z}_m - \mathbf{h}(\mathbf{x}_k)) \quad (2.35)$$

where $\mathbf{h}$ is the camera measurement function and $\mathbf{n}$ denotes the respective noises in operation. Utilizing these cost functions, we can define the optimization problem to recover all states at each time step from $\mathbf{x}_0$ to the current state $\mathbf{x}_k$. To solve this optimization problem, the cost functions need to be linearized at the current state estimate and can be solved using a Gauss-newton algorithm [\[91\]](#).

$$[\mathbf{x}_0^\top \quad \dots \quad \mathbf{x}_k^\top]^\top = \arg \min_{\mathbf{x}_0 \dots \mathbf{x}_k} \left(J_{\text{prior}} + \sum_{i=0}^k J_{\text{imu}}^i + \sum_{i=0}^k J_{\text{camera}}^i \right) \quad (2.36)$$

The factor graph optimization libraries GTSAM [\[31\]](#) and g2o [\[46\]](#) are two open-source libraries that are popular among researchers for building [VINS/VIO](#) algorithms. One of the key challenges in Batch least squares estimation is that the number of states can grow large, and the computational complexity can go beyond real-time implementation.

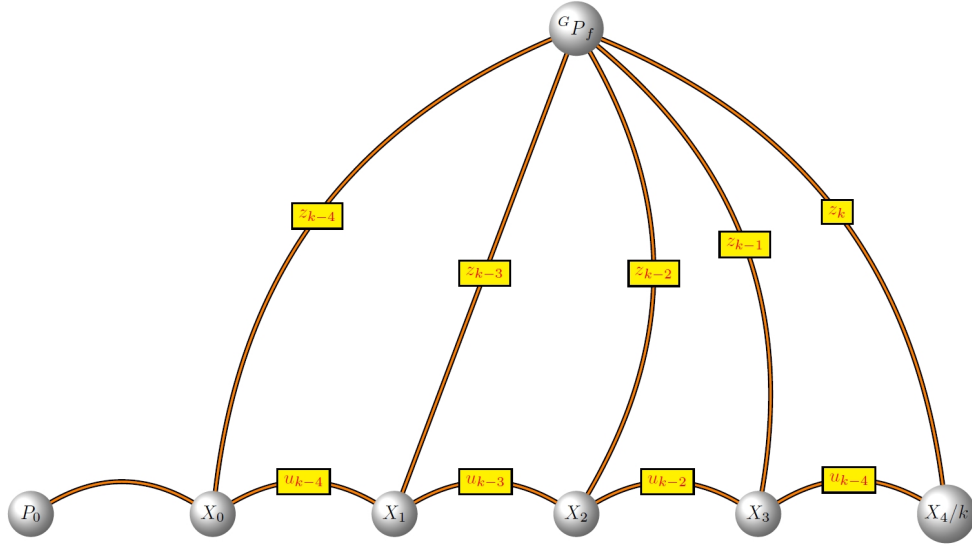


Figure 2.7: Factor graph form of visual-inertial state estimation.

2.3.5 Marginalization

Marginalization of the states is carried out for [VINS/VIO](#) algorithms to keep them close to computational complexity and improve efficiency. Some of the state-of-the-art methods for marginalization include fixed-lag smoother, sparsification, and incremental optimization.

The fixed-lag smoother [\[154\]](#) bounds the computational complexity by maintaining a fixed-size optimization window. This is achieved through marginalizing selected states. The sparsification methods [\[54\]](#) marginalize the state similar to the fixed-lag smoothers but further sparsifies the prior state that is to be added to the factor graph. In comparison to the aforementioned strategies, incremental optimization solvers such as iSAM2 [\[64\]](#) incrementally build up information and have heuristics and methodologies that allow for cost functions to not be re-evaluated at each time step. Instead, are only updated are only updated when directly involved with incoming measurements. This allows for large sparse graphs to be built that are efficient the majority of the time.

2.3.6 Inertial Pre-integration

As outlined in the previous topics, visual-inertial algorithms can obtain an accurate estimation of their state through nonlinear optimization. However, as explained before, real-time

optimization is a challenge when the trajectory increases. Accordingly, as the trajectory increases the number of variables in the optimization also grows with it. One of the main reasons for this issue is that the inertial measurements come at a much higher rate compared to the camera measurements. Hence, it increases the variables in the optimization much faster.

Examining Eq.(2.34) which gives the inertial cost function for estimating the next times step given the previous time step, if we use it as it is, we would need to reintegrate the function f_d at each time step. Hence, this can be computationally inefficient if the IMU is operating at a high frequency. The motivation of inertial pre-integration is to separate the state and inertial re-integration in order to avoid re-integrating f_d [39],[35].

2.3.7 Incremental Optimization

Incremental optimization takes advantage of an equivalent square-root information matrix form of batch least squares explained in Section 2.3.4. The standard batch least squares problem is given as

$$\arg \min_{\mathbf{x}} \|\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) - \mathbf{r}\|^2. \quad (2.37)$$

By taking the derivative of the quadratic cost function (2.37) to get the optimal update direction $\Delta \mathbf{x}$ we arrive at

$$\mathbf{H}^\top \mathbf{H} \Delta \mathbf{x} = \mathbf{H}^\top \mathbf{r}. \quad (2.38)$$

where, (2.38) is converted into an equivalent square root form through QR decomposition of $\mathbf{H}$ given as

$$\mathbf{H} = \mathbf{Q}\mathbf{U} \quad (2.39)$$

$$\mathbf{U}^\top \mathbf{U} \Delta \mathbf{x} = \mathbf{U}^\top \mathbf{Q}^\top \mathbf{r} \quad (2.40)$$

$$\mathbf{U} \Delta \mathbf{x} = \mathbf{Q}^\top \mathbf{r} \quad (2.41)$$

Then, (2.41) can be related back to the cost function (2.37) as

$$\arg \min_{\mathbf{x}} \|\mathbf{U}(\mathbf{x} - \hat{\mathbf{x}}) - \mathbf{Q}^\top \mathbf{r}\|^2. \quad (2.42)$$

The square root information matrix $\mathbf{U}$ here allows for very quick back substitution to solve for the state variables in (2.42). Here denoted $\mathbf{H}$ is the measurement jacobian, $\Delta \mathbf{x}$

is the optimal update direction obtained by the derivative of Eq.(2.37) and $\mathbf{r}$ denotes the residual terms. By taking the QR decomposition [128] of $\mathbf{H}$ we can construct an equivalent Eq.(2.41) to Eq.(2.38). Here, the matrix $\mathbf{Q}$ is an orthogonal matrix with $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}$ and $\mathbf{U}$ is an upper triangular matrix. This can then be related to the cost function as given in Eq.(2.42) by reformulating the original batch least square in square-root form. When a new measurement arrives at time step, k it is computationally more efficient to update the previous time step factorization $\mathbf{U}_{k-1}$ by directly QR-updating rather than updating and refactoring the whole matrix $\mathbf{H}$ again.

There exist several methods and variants of VINS algorithms which are built on incremental optimization. More briefly, Kejian et al. [147] proposed a square-root inverse sliding window filter that maintains the upper triangular Cholesky factor of the Hessian matrix of a sliding window of recent states. Micheal et al. [30] proposed incremental smoothing and mapping (iSAM) which utilizes the fast and incremental matrix factorization.

2.3.8 Observability Properties of Visual Inertial Navigation Systems

Understating the observability properties of VINS algorithms is another crucial aspect in estimator design for practical applications. These properties determine if it can recover the initial states given all collected measurements. It also provides insight into the minimum state parameters that are required to initialize the estimator. Furthermore, it is used to identify degenerate motions which lead to additional unobservable directions.

In order to perform observability analysis [153], typically the observability matrix is obtained. This matrix is denoted as

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \\ \vdots \\ \mathbf{M}_k \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \Phi_{(2,1)} \\ \vdots \\ \mathbf{H}_k \Phi_{k,1} \end{bmatrix}, \quad (2.43)$$

where $\mathbf{H}_k$ is the measurement jacobian at time step k , $\Phi_{(k,1)}$ is the state transition matrix from time-stamp t_1 to t_k .

This matrix $\mathbf{M}$ is a stacked matrix of measurement update jacobians and state transition matrices at each time step. Here, if we can find a matrix $\mathbf{N}$ such that $\mathbf{MN} = \mathbf{0}$ it indicates that we cannot uniquely identify the initial state purely based on the given

measurements. More formally $\mathbf{N}$ is the nullspace of $\mathbf{M}$ where it describes the unobservable directions.

As per the proofs given in [93, 52], for VINS algorithms that undergo generic 3D motion, there are 4DOF of unobservable directions. This consists of the 3DOF global translation and 1DOF rotation (yaw) around the gravity vector. Further explanation on this is given under section 2.3.10. Hence, these 4DOFs are not typically used for state initialization of VINS algorithms.

2.3.9 State Initialization

State initialization is the task of recovering the observable initial states of the system. For reliable state estimation of VINS algorithms, an accurate prior estimate (initial guess) is required. This is of importance for methods that use linearized measurement models, where the absence of an accurate initial estimate can lead to large linearization errors. As a result, this can cause divergence [33]. In particular, there exist several challenges that the state initialization procedure has to face in VINS algorithms. It has to be fast and robust, which needs to be done in a short period. The short window of time makes it difficult to recover the states.

The initialization process estimates the parameters scale, gravity direction, initial velocity, and accelerometer and gyroscope biases. These can be stacked and denoted as

$$\hat{\mathbf{x}}_0 = \left[({}^I\hat{\mathbf{q}}_{G,0})^\top \quad ({}^G\hat{\mathbf{p}}_{I,0}^G)^\top \quad ({}^G\hat{\mathbf{v}}_{I,0}^G)^\top \quad \hat{\mathbf{b}}_{g,0}^\top \quad \hat{\mathbf{b}}_{a,0}^\top \right]^\top. \quad (2.44)$$

The unobservable 4DOF state parameters ${}^G\hat{\mathbf{p}}_{I,0}^G$ and global yaw of ${}^I\hat{\mathbf{q}}_{G,0}$ can be set arbitrarily. The rest of the state parameters are recovered by state initialization. Although there exist many variants of methods for state initialization, some solutions to address this issue are briefly formulated below. The closed-form solutions [155, 94] are proposed by researchers with a tightly coupled approach. loosely coupled visual-inertial initialization is based on the assumption that up-to-scale camera trajectory can be estimated very accurately from pure monocular vision, where the inertial parameters can be estimated using this trajectory. This is the base idea on which the state-of-the-art ORB-SLAM [102] and VINS-mono [111] are built on. In contrast to this, in [17] proposed an efficient MAP estimator to obtain the inertial parameters and use these parameters to initialize visual parameters.

2.3.10 Long-term Navigation Key Problems

Despite the notable advancements in [VINS](#) algorithms, there remain certain challenges that need to be addressed for long-term navigation using [VINS](#). In particular, considering monocular [VINS](#) algorithms, there exist some inherent issues. First and foremost, let us consider monocular vision only. Firstly, with only monocular vision, it is not possible to recover metric scale. When a 3D scene is observed by a camera and re-projected into 2-D, the depth information is lost. Secondly, although scale estimation techniques have been developed the scale drift error is accumulated throughout, diminishing the localization accuracy [129]. Due to this reason, as the primary advantage [IMU](#) is utilized as an additional sensor to observe the metric scale as well as roll and pitch angles [111]. Furthermore, the integration of [IMU](#) increases the accuracy of the [VINS](#) system compensating for short-term visual feature loss, texture-less regions, motion blur, illumination changes, etc.

Elaborating further, when [VINS](#) is applied to ground vehicles whose motion is approximately planar and most of its motions are straight lines or uniform circular motions, there exist additional unobservable directions. As illustrated in [149] two such cases can be found:

1. Constant acceleration:
when a wheeled robot is moving on a straight line or circular arc with constant acceleration, the scale estimated by [VINS](#) becomes inaccurate. The argument here is that when the local acceleration is constant, one is unable to distinguish the true body acceleration magnitude from that of the accelerometer bias. This arbitrary nature of the true body acceleration results in scale ambiguity.
2. No rotation:
Similarly, when there is no rotation movement (typically a wheeled robot moving on a straight line) one is not able to differentiate between the local gravitational acceleration direction and the accelerometer bias resulting in roll and pitch ambiguity. Hence, in such cases, [VINS](#) estimates become inaccurate.

However, the visual, inertial and wheel encoder measurements are assumed under ideal conditions (i.e, no slippage, static environment etc.) and hence does not take into consideration the degradation of the odometry accuracy when fusing measurements prone to errors and outliers. Considering the above limitations, the subsequent chapters iterate the methodologies developed to correct faulty/error-prone measurements. Mainly, two major methodologies are developed: (a) [IMU](#) measurements are used to assist in eliminated dynamic visual feature points using a confidence based estimator; (b) Furthermore, to enhance the accuracy of the vehicle dynamics measurements

pre-integration (WO measurements), we propose to estimate the wheel slippage to compensate for the errors accumulating from WO using deep learning based approaches.

2.4 Visual Inertial Wheel Odometry

Due to the improvements in monocular VIO algorithms and low cost, they are being utilized in certain mobile robotic applications. Also, using a monocular camera has an advantage when building space-constrained miniature robots. However, there exist some inherent issues with monocular vision-based algorithms.

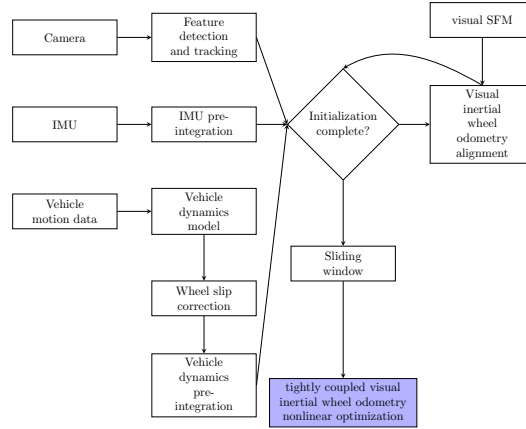


Figure 2.8: A typical VIWO system.

The motivation to use vehicle dynamic pre-integration is to provide alternative solutions for monocular VINS algorithm state estimation/initialization when the robot agent is undergoing degenerate motions. Degeneracy [152] describes situations/trajectories which cause additional unobservable directions for VINS estimation. In general, degenerate motions should be avoided as they diminish the system’s robustness. Furthermore, especially for monocular VINS algorithms, uniform linear or circular motions become scale and orientation (pitch and roll) unobservable [148]. Hence, directly applying VINS algorithms can provide inaccurate state estimates. The current state-of-the-art monocular VSLAM methods such as DSO [36], ORB-SLAM [102] can create a sparse track-able map but suffer from accurate estimation of pitch, roll, and absolute scale. Considering these fallbacks, we utilize the vehicle dynamic measurements to aid in robust initialization. Through the pre-integration of the vehicle dynamic measurements, we obtain sufficient translation information as well as information about wheel slippages.

The fusing of **VIO** with **WO** is a challenging task. This is because under certain conditions due to wheel slippages, the **WO** measurements can be faulty. Secondly, the combined system would need a good initialization, which is a crucial aspect for accurate localization. Furthermore, by pre-integration of **VIO** the nonlinear optimization becomes heavy due to the additional information. Hence, the motivation of this study is to propose an integrative approach to tackle these problems sequentially. Although **IMU** and **WO** can aid **VO** for better performance in **VIO** problem, they still need to be optimized and refined together to achieve superior performance which can be comparable with state of the art methods. Furthermore, to improve the pre-integration accuracy of **WO** we introduce wheel slip compensation for mobile wheeled robots.

Chapter 3

Comparative Numerical Analysis of State of the Art VINS Algorithms

This chapter aims to investigate some existing state-of-the-art [VINS](#) algorithms. The performance of each of the algorithms is recorded to create a baseline for bench-marking the algorithms proposed in the subsequent chapters. These state-of-the-art algorithms are run on certain challenging datasets to demonstrate the potential areas of improvement. It was also observed that there exists a scarcity of datasets for [SLAM](#) system evaluation for wheeled mobile robots.

All the algorithms are run using ROS on Linux platform. The comparison is carried out by calculating the standard error metrics to calculate the accuracy of the algorithms. Here, we investigate the state-of-the-art ORBSLAM3 [\[16\]](#), and VINS-Fusion (which is based on VINS-mono [\[111\]](#)). Furthermore, ORBSLAM2 [\[103\]](#) is added to the comparison to compare the accuracy with respect to [VSLAM](#) algorithms.

3.1 Simulation Setup

Calibration is one of the crucial components of [VINS/VIO](#) algorithms. These algorithms have many calibration parameters such as camera-inertial, spatial-temporal parameters, camera intrinsic parameters (focal length, center, dist), and inertial intrinsic (scale, skew) parameters. The time instants at which the inertial and camera sensors provide information are crucially important for the accuracy and robustness of the algorithms. In practical scenarios, temporal misalignment between these sensors occur due to transmission delays

[114]. The calibration process can be performed offline prior to state estimation, or online during state estimation, or both. However, offline calibration is highly accurate compared to online calibration. However, online calibration provides more flexibility in practical deployments. Additionally, it can handle time-varying calibration parameters.

3.1.1 Metrics and Evaluation

There exist several metrics to compare different VO/VIO algorithms [159]. The absolute trajectory error (ATE) given as

$$e_{ATE} = \frac{1}{N} \sum_{i=0}^N \sqrt{\frac{1}{K} \sum_{k=1}^K \|\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,i}^+\|_2^2}, \quad (3.1)$$

is defined as the average RMSE (root mean squared error) of the 3D position of the robot at each time step after performing trajectory alignment. Here e_{ATE} is obtained by taking the average of N number of runs. The relative pose error (RPE) is defined as

$$e_{RPE,d_i} = \frac{1}{D_i} \sum_{k=1}^{D_i} \|\tilde{\mathbf{x}}_r - \hat{\mathbf{x}}_r\|_2^2, \quad (3.2)$$

where; $\hat{\mathbf{x}}_r = \mathbf{x}_k - \mathbf{x}_{k+d_i}$ and d_i is a specific trajectory length. The normalized estimation error squared (NEES) is given as,

$$e_{NEES,k} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,i})^\top \mathbf{P}_{k,i}^{-1} (\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,i}). \quad (3.3)$$

Before calculating these error metrics one should carefully perform trajectory alignment as the ground truth and estimated trajectories are usually expressed in different reference frames. A standard method to achieve this is Umeyama's method [141].

3.2 Simulation Studies

In this section, the current state-of-the-art VINS algorithms are tested through simulation studies on real-world data. Table 3.1 shows simulation results carried out on EuRoc dataset [13].

sequence	ORB SLAM3		ORB-SLAM2	ORB-SLAM	VINS-Fusion	VINS-mono
	Stereo-VIO	Mono-VIO	Stereo-VO	Mono-VO	Stereo-VIO	Mono-VIO
MH 01	0.029	0.062	0.035	0.071	0.540	0.084
MH 02	0.033	0.037	0.018	0.067	0.460	0.105
MH 03	0.035	0.046	0.028	0.071	0.330	0.074
MH 04	0.051	0.075	0.119	0.082	0.780	0.122
MH 05	0.082	0.057	0.060	0.060	0.500	0.147
VR 1 01	0.038	0.049	0.035	0.015	0.550	0.047
VR 1 02	0.014	0.015	0.020	0.020	0.230	0.066
VR 1 03	0.024	0.037	0.048	-	-	0.180
VR 2 01	0.032	0.042	0.037	0.021	0.230	0.056
VR 2 02	0.014	0.021	0.035	0.018	0.200	0.090
VR 2 03	0.024	0.027	-	-	-	0.244

Table 3.1: Simulation study EuRoc dataset.

where; MH stands for Machine hall and VR stands for Vicon room.

Overall ORB-SLAM3 provided the best results in terms of accuracy([ATE](#)) among the state of the art algorithms compared. Hence, it is proposed to take ORB-SLAM3 as a benchmark to evaluate proposed algorithms in the subsequent chapters. The following figures depict the trajectory, drift and error characteristics for ORB-SLAM3.

The TUM dataset [\[131\]](#) includes data scenes with dynamic objects. ORB-SLAM3 is again evaluated using TUM dynamic data sequences with dynamic objects to understand the effect of dynamic objects as well as to how well the state of the art algorithm is capable of handling these dynamic objects. The results illustrated in Table [3.2](#) show that there is a significant decrease in accuracy compared to the EuRoc dataset, where the scene is static. The average error of EuRoc dataset with ORB-SLAM 3 was 0.034 whereas in the average error increased to 0.44492 for the TUM dynamic dataset. Also, an important thing to note here is that these results are obtained along with loop closure corrections. In real world scenarios, mobile robots undergo many different paths where loop closure is not always possible. Hence, the effect of dynamic objects are diluted. Nevertheless, this shows that more research is required to design [VISLAM/VSLAM](#) algorithms which are capable of handling realistic scenarios for the advancement of wheeled mobile robots.

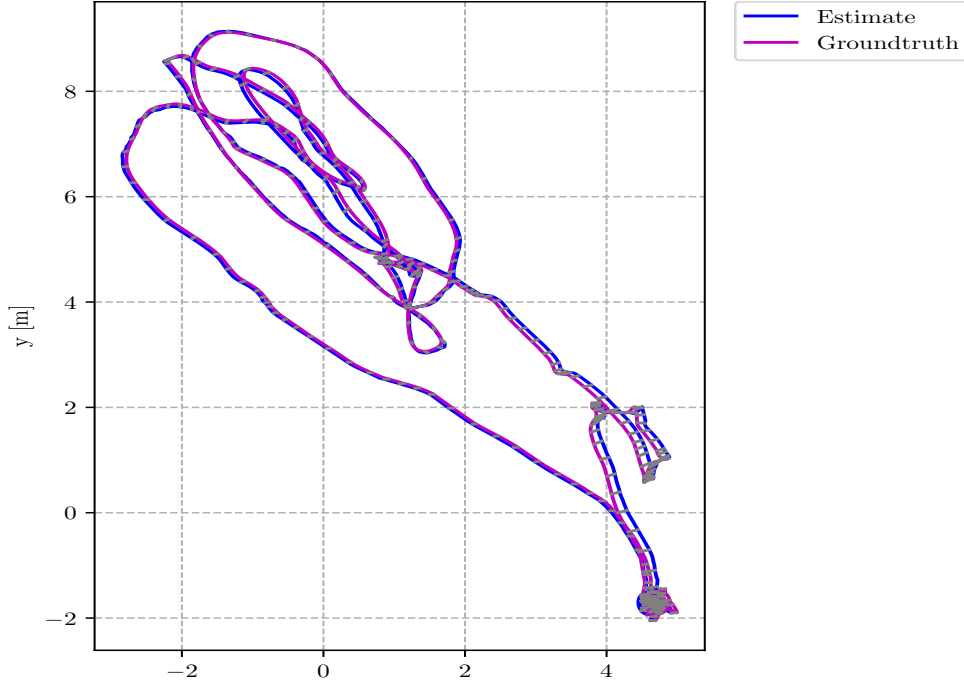


Figure 3.1: ORB-SLAM-3 on Euroc mh-01 trajectory top view.

3.3 Drawbacks of the Current State of the Art Methods

Although ORB-SLAM3 is capable of providing very accurate performance with loop closure, there still exist some challenges which need to be addressed for robust performance. As outlined in [16] in cases where the motions are very slow or without roll and pitch it is extremely challenging for ORB-SLAM3 to initialize the IMU. The initialization errors can have a big impact on the accuracy of VINS algorithms. As an example, this can be a wheeled mobile robot navigating on a flat surface. Hence, Chapter 4 proposes to utilize the additional sensory data coming from wheel encoders as an aid to mitigate these issues.

Furthermore, many feature-based VISLAM algorithms like ORB-SLAM 3 use RANSAC to remove outliers in the visual motion estimation step. This is the step where the algorithm is able to filter out wrong data associations arising from dynamic objects. For RANSAC to be accurate, it requires the majority of correspondence points to be static with respect to the camera motion. Hence, when the dynamic objects cover a majority of the field of view of the camera and the majority of visual features lie in these dynamic objects,

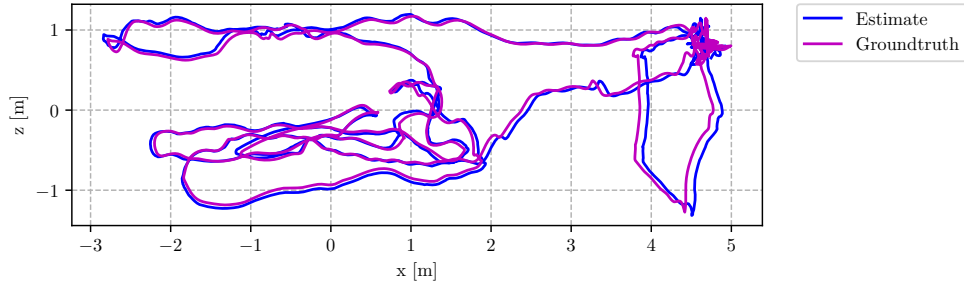
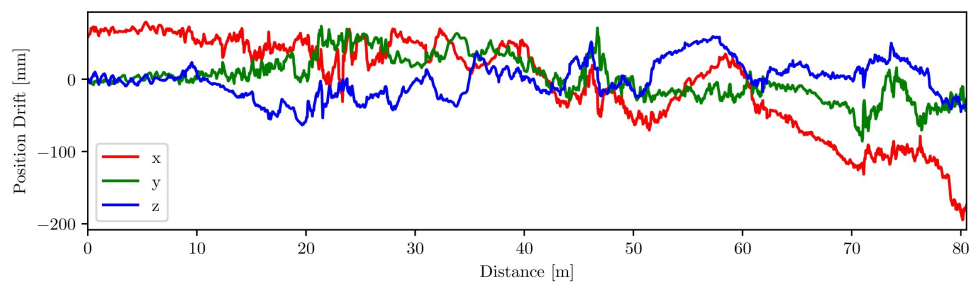


Figure 3.2: ORB-SLAM-3 on Euroc mh-01 trajectory side view.

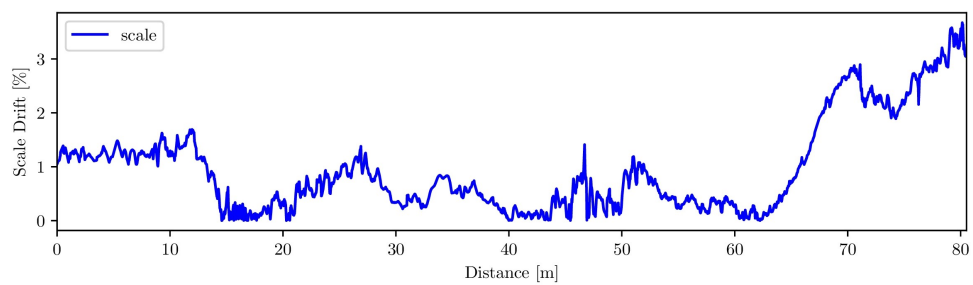
sequence	ORB-SLAM-3
fr3-walking-xyz	0.6687
fr3-walking-rpy	0.8461
fr3-walking-static	0.1072
fr3-walking-halfsphere	0.5939
fr3-sitting-static	0.0087

Table 3.2: Simulation results for ORB-SLAM-3 using TUM dynamic dataset.

the feature-based [SLAM](#) approach can fail [\[11\]](#). Due to this reason, alternative methods for removing outliers from dynamic objects need to be surveyed. In addition, when these type of [VSLAM](#)/[VISLAM](#) algorithms are deployed to wheeled mobile robots the front end localization modules does not use available valuable sensory information such as wheel encoder measurements to optimize the odometry and localization accuracy.

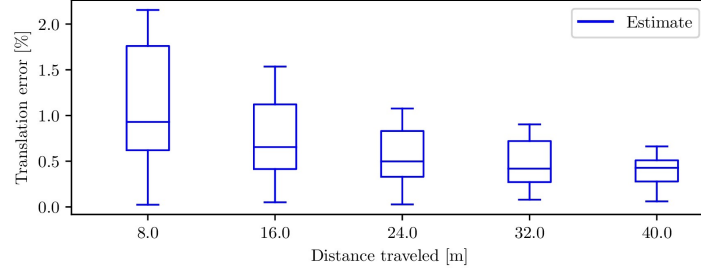


(a) Position drift

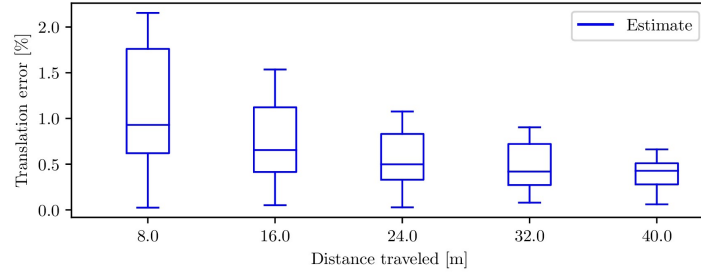


(b) Scale drift

Figure 3.3: ORB-SLAM-3 on Euroc mh-01 trajectory position and scale drift.



Relative translation error



Relative translation error percentage

Figure 3.4: ORB-SLAM-3 on Euroc dataset machine hall 01 sequence translation error and percentage.

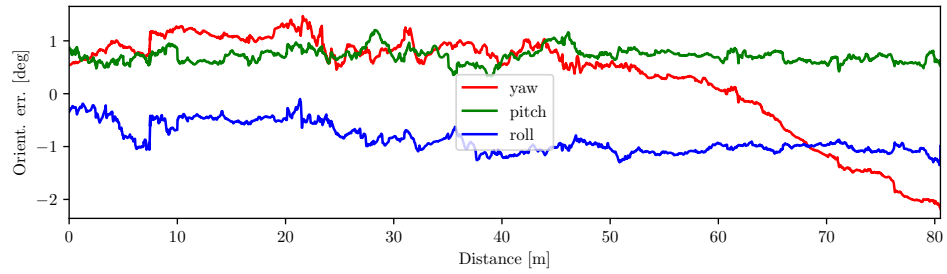


Figure 3.5: ORB-SLAM-3 on Euroc dataset machine hall 01 sequence rotation error.

Chapter 4

Dynamic Feature Point Elimination

Feature-based [VSLAM](#) front end modules mostly use random sample consensus [RANSAC](#) as a standard outlier removal method when estimating ego-motion. However, [RANSAC](#) does not work well when the majority of the feature points are dynamic or when the majority of the field of view of the camera is covered by a dynamic object [78]. If the environment is dynamic object dominant, the features that remained after dynamic object/feature elimination might not be sufficient for the feature matching process of [RANSAC](#). As an alternative to [RANSAC](#), some of the current [VSLAM](#) applications use neural networks or image processing/segmentation techniques, to identify possible dynamic objects, and remove the feature points associated with them. However, neural network-based approaches suffer from training data-set limitations, which affect the accuracy of dynamic feature (outlier) detection and elimination. Hence, an alternative approach is required to remove the dynamic object feature points as outliers.

The majority of [VSLAM](#) algorithms in the literature use camera position estimation by decomposing the fundamental matrix that relates two subsequent images obtained by the camera attached to the robot [59]. The trifocal tensor plays a similar role to the fundamental matrix, relating to three subsequent images obtained by the robot. It encapsulates all the geometric relations between the three adjacent views that are independent of scene structure [48]. Compared to the fundamental matrix, the trifocal tensor also relates lines in three adjacent views seen by the robot camera. The results can be typically improved by using lines and points for feature extraction [157, 72, 156]. Although the existing applications of trifocal tensor techniques to [VINS](#) are limited [55, 137, 75], when considering three views there are additional stronger constraints for estimating the trifocal tensor than the fundamental matrix making the trifocal tensor technique advantageous [137].

Although it is the main outlier removal mechanism used in both fundamental matrix and trifocal tensor estimation, **RANSAC** is not able to remove the noise entirely when dynamic feature points overwhelm static feature points [69]. Hence, the **RANSAC**-based outlier rejection schemes need further filtering mechanisms. Aligned with this need, in this paper, we propose a confidence estimation-based method to eliminate dynamic feature points by utilizing the **IMU** data and distinguishing static and dynamic feature points. In the proposed method, when the confidence in **IMU** data is lower than a defined threshold value, the outlier rejection scheme is switched back to pure **RANSAC**-based outlier removal.

The main contributions of this chapter are (i) the design of an **IMU**-data-based confidence level estimator to eliminate dynamic feature points and (ii) a multi-state constraint Kalman filter (**MSCKF**) framework design to estimate robot position and orientation using a visual, inertial and wheel encoder measurement data. The chapter is organized as follows. The problem definition and system description are provided in Section 4.1. The proposed method is explained in Section 4.2. The robot odometry estimation utilizing **MSCKF** is included in Section 4.3. In Section 4.4, the outlier removal utilizing **IMU** data based confidence estimation integrated with **RANSAC** for estimating trifocal tensor is explained. In Section 4.5, experimental results are provided and discussed based on real world data. Finally, the concluding remarks are provided in Section 4.6.

4.1 The System and Problem Definition

In system formulation, the superscript $^G(\cdot)$ denotes the global frame, $^I(\cdot)$ the body-fixed **IMU** frame, $^C(\cdot)$ the camera frame, and $^O(\cdot)$ the wheel-odometry (**WO**) measurement frame with the origin at the center of the axle connecting the two wheels of the robot. For position, velocity, and acceleration, the upper postscript denotes the frame that the quantity is relative to. For example, $^G(\cdot)_O$ denotes wheel-odometry's frame with respect to the global frame. $^G(\cdot)_O^I$ denotes a quantity (such as position, velocity, etc.) of the wheel odometry frame relative to the **IMU** frame, but expressed in the global frame. An orientation of a frame can be expressed by either a unit quaternion $\mathbf{q}$ or a rotation matrix $\mathbf{C}$. $\mathbf{o}_k$ denotes the pose of the k^{th} **WO** frame when taking the k^{th} image. The symbol $\otimes$ denotes the multiplication operation between two quaternions. ${}^C\mathbf{p}_I^C, {}^C\mathbf{q}_I^C$ are the **IMU** extrinsic parameters relative to the camera, ${}^O\mathbf{p}_C^O, {}^O\mathbf{q}_C^O$ are the extrinsic parameters between the camera and **WO** frames, and ${}^I\mathbf{p}_O^I, {}^I\mathbf{q}_O^I$ are the extrinsic parameters between **WO** and **IMU** frames.

4.1.1 Robot State Vector

At each sampled time $t = t_k = kT_s$ (where T_s is the sampling time) the state vector $\mathbf{x}_k = [\mathbf{x}_{I,k}^\top \quad \mathbf{x}_{C,k}^\top]^\top$ of the robot consists of the current inertial state

$$\mathbf{x}_{I,k} = [{}^{I_k}\mathbf{q}_G^\top \quad ({}^G\mathbf{p}_{I_k}^\top)^\top \quad ({}^G\mathbf{v}_{I_k}^\top)^\top \quad \mathbf{b}_g^\top \quad \mathbf{b}_a^\top]^\top, \quad (4.1)$$

and the lumped vector of the previous n world to inertial frame transformations,

$$\mathbf{x}_{C,k} = [{}^{I_{k-1}}\mathbf{q}_G^\top \quad ({}^G\mathbf{p}_{I_{k-1}}^\top)^\top \quad \dots \quad {}^{I_{k-n}}\mathbf{q}_G^\top \quad ({}^G\mathbf{p}_{I_{k-n}}^\top)^\top]^\top, \quad (4.2)$$

where ${}^{I_j}\bar{\mathbf{q}}_G$ denotes the quaternion corresponding to the rotation ${}^{I_j}\mathbf{C}_G$ and ${}^G\mathbf{p}_{I_j}^\top$ denotes the translation vector from global frame $\{G\}$ to inertial body-fixed frame $\{I\}$ at step j , for $j = k - n, \dots, k$. ${}^G\mathbf{p}_{I_k}^\top, {}^G\mathbf{v}_{I_k}^\top$ are the position and velocity of the frame $\{I\}$ relative to and with respect to the frame $\{G\}$, and $\mathbf{b}_g, \mathbf{b}_a$ are the gyroscope and accelerometer biases respectively,

The inertial state $\mathbf{x}_{I,k}$ is propagated forward using IMU kinematics. At time t_k , the measurement model from the gyroscope $\boldsymbol{\omega}_m$ and the measurement model from the accelerometer $\mathbf{a}_m$ are obtained as [20]

$$\boldsymbol{\omega}_{m,k} = {}^{I_k}\boldsymbol{\omega}_{I_k}^\top + \mathbf{b}_{g,k} + \mathbf{n}_{g,k}, \quad (4.3)$$

$$\mathbf{a}_{m,k} = {}^{I_k}\mathbf{a}_{I_k}^\top + {}^{I_k}\mathbf{C}_G ({}^{I_k}\mathbf{q}_G)^\top {}^G\mathbf{g} + \mathbf{b}_{a,k} + \mathbf{n}_{a,k}, \quad (4.4)$$

where $\mathbf{n}_g, \mathbf{n}_a$ are the considered zero-mean Gaussian noises.

4.1.2 Camera Measurements and Visual Odometry

The bearing vector of feature i measured by camera at time step k , with camera projection $\boldsymbol{\pi}([x, y, z]^\top) = [\frac{x}{z}, \frac{y}{z}]^\top$, is

$$\mathbf{z}_{m,k}^i = \boldsymbol{\pi}({}^{C_k}\mathbf{p}_{f_i}^{C_k}) + \mathbf{n}_{i,k}, \quad (4.5)$$

$${}^{C_k}\mathbf{p}_{f_i}^{C_k} = {}^C\mathbf{C}_I {}^{I_k}\mathbf{C}_G ({}^{I_k}\mathbf{q}_G) \cdot ({}^G\mathbf{p}_{f_i}^\top - {}^G\mathbf{p}_{I_k}^\top) + {}^C\mathbf{p}_I^C, \quad (4.6)$$

where ${}^G\mathbf{p}_{f_i}^G$ is the i^{th} 3D feature position, $\mathbf{n}_{i,k}$ is the measurement noise of the i^{th} feature, ${}^C\mathbf{C}_I, {}^C\mathbf{p}_I^C$ are the camera to IMU extrinsic information. Filtered measurement estimates $\hat{\mathbf{z}}_{m,k}^i$ and the state estimate $\hat{\mathbf{x}}_k$ are obtained by Kalman filtering based on the residuals $\tilde{\mathbf{z}}_{m,k}^i, \tilde{\mathbf{x}}_k$, and the process model,

$$\tilde{\mathbf{z}}_{m,k}^i = \mathbf{H}_{x_i} \tilde{\mathbf{x}}_k + \mathbf{H}_{f_i} {}^G\tilde{\mathbf{p}}_{f_i}^G + \mathbf{n}_{i,k} \quad (4.7)$$

where

$$\begin{aligned} \tilde{\mathbf{z}}_{m,k}^i &= \mathbf{z}_{m,k}^i - \hat{\mathbf{z}}_{m,k}^i \\ \tilde{\mathbf{x}}_k &= \mathbf{x}_k - \hat{\mathbf{x}}_k \end{aligned}$$

$\mathbf{H}_{x_i}$ is the robot state Jacobian, and $\mathbf{H}_{f_i}$ is the feature position Jacobian.

Stacking the residuals $\tilde{\mathbf{z}}_{m,k}^i$ for all feature points i and applying nullspace marginalization based on (4.7), a camera measurement residual process model independent of the feature positions is obtained as,

$$\tilde{\mathbf{z}}_{m,k}^i = \mathbf{H}_x \tilde{\mathbf{x}}_k + \mathbf{n}_{i,k}, \quad (4.8)$$

which is directly used in the MSCKF design to estimate the robot states as in [76]. Since the feature positions are not stored in the state vector, this results in high computational efficiency.

4.1.3 Wheel Encoder Measurements

Wheel encoder pre-integration is used to provide alternative robust solutions for monocular VINS-based state estimation/initialization when the robot agent is undergoing degenerate motions, i.e. situations/trajectories which cause additional unobservable directions for VINS-based state estimation [152]. Especially for monocular VINS algorithms when the robot is undergoing uniform linear or circular motions, the scale and orientation (pitch and roll) become unobservable [148]. For wheeled autonomous mobile robots, the wheel angle and angular rate measurements can be used to measure the longitudinal speed v and angular velocity ω of the robot's center of gravity in the body frame and assist VINS-based state estimation when the robot undergoes degenerate motions.

Since the WO measurements arrive at a very high rate compared to the camera measurements, it would be computationally too expensive to perform the MSCKF update at that rate. Hence, they are pre-integrated over the time of the most recent two imaging time intervals t_k and t_{k+1} as

$$\mathbf{o}_{k+1} = [\theta_{k+1} \quad x_{k+1} \quad y_{k+1}]^\top = \int_{\tau=t_k}^{t_{k+1}} [\omega_z \quad v \cos(\theta_k) \quad v \sin(\theta_k)]^\top d\tau, \quad (4.9)$$

where θ is the yaw angular difference between the global frame and **WO** measurements frame, and v is the longitudinal velocity of the robot in **WO** frame.

4.1.4 IMU Measurements

To estimate the fundamental matrix accurately using IMU data, the IMU biases $\mathbf{b}_a$ and $\mathbf{b}_g$ need to be estimated properly at the initialization stage, utilizing visual, inertial and **WO** information. The **IMU** provides pre-integrated translation, velocity, and rotation in quaternion between the consecutive IMU-attached frames I_k to I_{k+1} denoted as ${}^{I_k}\delta\mathbf{p}_{I_{k+1}}^{I_k}$, ${}^{I_k}\delta\mathbf{v}_{I_{k+1}}^{I_k}$, and ${}^{I_k}\delta\mathbf{q}_{I_{k+1}}^{I_k}$ respectively. We refer the reader to [112] for more information regarding ${}^{I_k}\delta\mathbf{p}_{I_{k+1}}^{I_k}$, ${}^{I_k}\delta\mathbf{v}_{I_{k+1}}^{I_k}$, and ${}^{I_k}\delta\mathbf{q}_{I_{k+1}}^{I_k}$. The vehicle dynamics position and rotation in the world frame is denoted respectively as

$$\mathbf{P}_{v_{k+1}}^w = \mathbf{P}_{v_k}^w + \int_{t \in [t_k, t_{k+1}]} [\mathbf{R}_t^w + (\hat{\mathbf{v}}_t - \mathbf{n}_v)] dt, \quad (4.10)$$

$$\mathbf{q}_{v_{k+1}}^w = \mathbf{q}_{v_k}^w \otimes \int_{t \in [t_k, t_{k+1}]} \frac{1}{2} \Omega(\hat{\boldsymbol{\omega}}_t - \mathbf{n}_\omega) \mathbf{q}_t^{v_k} dt, \quad (4.11)$$

where; $\Omega(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & \hat{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & \omega_x & 0 \end{bmatrix}$, $\mathbf{n}$ denotes the measurement noise modelled as zero mean white Gaussian noise.

This is then converted to the respective continuous time pre-integration terms for the time period $[t_k, t_{k+1}]$ as

$$\boldsymbol{\alpha}_{v_{k+1}}^{v_k} = \int_{t \in [t_k, t_{k+1}]} [\mathbf{R}_t^w + (\hat{\mathbf{v}}_t - \mathbf{n}_v)] dt, \quad (4.12)$$

$$\boldsymbol{\gamma}_{v_{k+1}}^{v_k} = \int_{t \in [t_k, t_{k+1}]} \frac{1}{2} \Omega(\hat{\boldsymbol{\omega}}_t - \mathbf{n}_\omega) \boldsymbol{\gamma}_t^{v_k} dt, \quad (4.13)$$

where $\boldsymbol{\alpha}_{v_{k+1}}^{v_k}, \boldsymbol{\gamma}_{v_{k+1}}^{v_k}$ are the position and rotation preintegration terms.

4.1.5 Scale Estimation

Scale estimation is an important step, especially in monocular **VIO** systems. The following steps summarize the methodology for scale estimation.

By aligning the dynamic measurements, it is possible to estimate the metric scale. Using the vision only **SFM** as explained in Section 2.3.10, it is only possible to estimate an arbitrary scale of the camera trajectory. Due to this reason, we introduce scale factor s when transforming the coordinate system between the camera and vehicle dynamic system as

$$s {}^{C_0}\mathbf{p}_{O_k}^{C_0} = s {}^{C_0}\mathbf{p}_{C_k}^{C_0} - {}^{C_0}\mathbf{C}_{O_k} {}^O\mathbf{p}_C^O. \quad (4.14)$$

Combining Eq.'s (4.10),(4.11),(4.12),(4.13),(4.14) we get

$${}^O\mathbf{C}_I {}^{I_k}\mathbf{C}_{C_0} ({}^{C_0}\mathbf{p}_{C_{k+1}}^{C_0} - {}^{C_0}\mathbf{p}_{C_k}^{C_0})s = {}^{O_k}\delta\hat{\mathbf{p}}_{O_{k+1}} + {}^O\mathbf{C}_I {}^{I_k}\mathbf{C}_{C_0} {}^{C_0}\mathbf{C}_{I_{k+1}} {}^I\mathbf{C}_O {}^O\mathbf{p}_C^O - {}^O\mathbf{p}_C^O \quad (4.15)$$

$$\mathbf{R}_b^v \mathbf{R}_{c_0}^{b_k} (\bar{\mathbf{P}}_{c_{k+1}}^{c_0} - \bar{\mathbf{P}}_{c_{k+1}}^{c_0})s = \hat{\alpha}_{v_{k+1}}^{v_k} + \mathbf{R}_b^v \mathbf{R}_{c_0}^{b_k} \mathbf{R}_{b_{k+1}}^{c_0} \mathbf{R}_v^b p_c^v - p_c^v.$$

writing the above equation in a simplified form we can arrive at

$${}^{O_k}\mathbf{L}_{O_{k+1}} s = {}^{O_k}\hat{\mathbf{z}}_{O_{k+1}}. \quad (4.16)$$

Now the scale is estimated by solving the linear least squares problem given by

$$\min_s \sum_{k \in W} \| {}^{O_k}\hat{\mathbf{z}}_{O_{k+1}} - {}^{O_k}\mathbf{L}_{O_{k+1}} s \|^2, \quad (4.17)$$

where W denotes the sliding window.

4.1.6 Gyroscope Bias Estimation

Once the pre-integrated IMU measurements are obtained, they are used with the WO pre-integrated measurements to estimate the gyroscope bias $\mathbf{b}_g$. We minimize the pre-integration [21] between the WO rotational difference ${}^{O_{k+1}}\delta\mathbf{q}_{O_k}$ and gyroscope rotational difference ${}^{I_k}\delta\mathbf{q}_{I_{k+1}}$, by finding the least squares solution to minimization of

$$\min_{\delta \mathbf{b}_g} \sum_{k \in W} \left\| {}^I \mathbf{q}_O \otimes ({}^{O_{k+1}} \delta \mathbf{q}_{O_k}) \otimes {}^I \mathbf{q}_O^{-1} \otimes {}^I \delta \mathbf{q}_{I_{k+1}} \right\|^2, \quad (4.18)$$

where

$$\begin{aligned} {}^I \delta \mathbf{q}_{I_{k+1}} &= {}^I \delta \hat{\mathbf{q}}_{I_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \mathbf{J}_{\mathbf{b}_g}^q \delta \mathbf{b}_g \end{bmatrix} \\ \mathbf{J}_{\mathbf{b}_g}^q &= \frac{\partial {}^I \delta \mathbf{q}_{I_{k+1}}}{\partial \mathbf{b}_g}, \end{aligned}$$

and ${}^{O_{k+1}} \delta \mathbf{q}_{O_k}$ is the rotation pre-integration term in **WO** frame $\{\mathbf{O}\}$ obtained using the wheel encoders, and $k \in W$ indicates all frames k in the sliding window W .

4.1.7 Accelerometer Bias Estimation

Once the scale and gyroscope bias $\mathbf{b}_g$ are estimated, all the three data streams incoming from the camera, **IMU**, and robot **WO** measurements are used to estimate the velocity $\mathbf{v}$, gravity vector $\mathbf{g}$, and accelerometer bias $\mathbf{b}_a$. We also note that when the robot does not undergo rotation motion it is difficult to estimate the accelerometer bias in the initialization procedure [113]. Also, from a practical aspect, it does not make sense for a robot to make aggressive rotation movements at the initialization phase. We also need to estimate the gravity vector to align the obtained odometry with the world coordinate system. However, this observability issue can be mitigated by introducing the WO pre-integration. Therefore, the variables to be estimated are denoted as

$$\boldsymbol{\chi}_k = [({}^I \mathbf{v}_{I_k}^{C_0})^\top, ({}^{I_{k+1}} \mathbf{v}_{I_{k+1}}^{C_0})^\top, {}^{C_0} \mathbf{g}^\top, \delta \mathbf{b}_{a,k}^\top]^\top, \quad (4.19)$$

where ${}^I \mathbf{v}_{I_j}^{C_0}$ denotes the linear velocity in IMU frame at step j relative to the first camera frame, ${}^{C_0} \mathbf{g}$ is gravity in reference camera frame, $\delta \mathbf{b}_{a,k} \approx \mathbf{b}_{a,k} - \mathbf{b}_{a,k-1}$ is the accelerometer bias increment. Between two consecutive frames I_k and I_{k+1} , we have [112, 65]

$$\begin{aligned} {}^I \delta \mathbf{p}_{I_{k+1}}^{I_k} &= {}^I \mathbf{C}_{C_0} \left[({}^{C_0} \mathbf{p}_{I_{k+1}}^{C_0} - {}^{C_0} \mathbf{p}_{I_k}^{C_0}) + \frac{1}{2} {}^{C_0} \mathbf{g} T_s^2 - {}^{C_0} \mathbf{C}_{I_k} {}^I \mathbf{v}_{I_k}^{C_0} T_s \right] \approx {}^I \delta \hat{\mathbf{p}}_{I_{k+1}}^{I_k} + \mathbf{J}_{\mathbf{b}_a}^p \delta \mathbf{b}_{a,k} \\ {}^I \delta \mathbf{v}_{I_{k+1}}^{I_k} &= {}^I \mathbf{C}_{C_0} \left[{}^{C_0} \mathbf{C}_{I_{k+1}} {}^{I_{k+1}} \mathbf{v}_{I_{k+1}}^{C_0} + {}^{C_0} \mathbf{g} T_s - {}^{C_0} \mathbf{C}_{I_k} {}^I \mathbf{v}_{I_k}^{C_0} \right] \approx {}^I \delta \hat{\mathbf{v}}_{I_{k+1}}^{I_k} + \mathbf{J}_{\mathbf{b}_a}^v \delta \mathbf{b}_{a,k}, \end{aligned} \quad (4.20)$$

and the translation of body frame and WO frame is given as

$${}^{C_0}\mathbf{p}_{I_{k+1}}^{C_0} - {}^{C_0}\mathbf{p}_{I_k}^{C_0} = {}^I\mathbf{C}_O {}^I\mathbf{C}_{C_0} {}^I\mathbf{C}_{C_0} {}^I\delta\hat{\mathbf{p}}_{I_{k+1}} + ({}^{C_0}\mathbf{C}_{I_{k+1}} {}^I\mathbf{C}_O - {}^{C_0}\mathbf{C}_{I_k} {}^I\mathbf{C}_O) {}^O\mathbf{p}_I. \quad (4.21)$$

$$\mathbf{p}_{b_{k+1}}^{c_0} - \mathbf{p}_{b_k}^{c_0} = \mathbf{R}_v^b \mathbf{R}_{c_0}^{b_k} \hat{\alpha}_{b_{k+1}}^{b_k} + (\mathbf{R}_{b_{k+1}}^{c_0} \mathbf{R}_v^b - \mathbf{R}_{b_k}^{c_0} \mathbf{R}_v^b) \mathbf{p}_b^v. \quad (4.22)$$

By combining Eqs. (4.20), (4.21), and (4.22) a linear measurement model is obtained as

$$\hat{\mathbf{z}}_{I,k} = \begin{bmatrix} \hat{\mathbf{z}}_{I_1,k} \\ \hat{\mathbf{z}}_{I_2,k} \end{bmatrix} = {}^I\mathbf{G}_{I_{k+1}} \boldsymbol{\chi}_k + {}^I\mathbf{n}_{I_{k+1}}. \quad (4.23)$$

where

$$\hat{\mathbf{z}}_{I_1,k} = {}^I\mathbf{C}_{C_0} ({}^{C_0}\mathbf{p}_{I_{k+1}}^{C_0} - {}^{C_0}\mathbf{p}_{I_k}^{C_0}),$$

$$\hat{\mathbf{z}}_{I_2,k} = {}^I\mathbf{C}_{C_0} \delta\hat{\mathbf{v}}_{I_{k+1}}^{I_k},$$

$${}^I\mathbf{G}_{I_{k+1}} = \begin{bmatrix} -\mathbf{I}T_s & \mathbf{0} & \frac{1}{2} {}^I\mathbf{C}_{C_0} T_s^2 & -\mathbf{J}_{\mathbf{b}_a}^{\mathbf{p}} \\ -\mathbf{I} & {}^I\mathbf{C}_{C_0} {}^{C_0}\mathbf{C}_{I_{k+1}} & {}^I\mathbf{C}_{C_0} T_s & -\mathbf{J}_{\mathbf{b}_a}^{\mathbf{v}} \end{bmatrix},$$

and ${}^I\mathbf{n}_{I_{k+1}}$ are the stacked measurement noises. This is solved again as a linear least squares problem of minimizing

$$\min_{\boldsymbol{\chi}} \sum_{k \in W} \|\hat{\mathbf{z}}_{I,k} - {}^I\mathbf{G}_{I_{k+1}} \boldsymbol{\chi}_k\|^2. \quad (4.24)$$

where

$$\boldsymbol{\chi} = [({}^{I_0}\mathbf{v}_{I_0}^{C_0})^\top, \dots, ({}^{I_n}\mathbf{v}_{I_n}^{C_0})^\top, {}^{C_0}\mathbf{g}^\top, \delta\mathbf{b}_{a,0}^\top, \dots, \delta\mathbf{b}_{a,n}^\top]^\top,$$

Once the variables are estimated, the rotation ${}^G\mathbf{q}_{C_0}$ between the global frame and camera frame C_0 is obtained by rotating the gravity to the z-axis. Then all the variables are rotated from the reference frame ${}^{C_0}(\cdot)$ to the global frame ${}^G(\cdot)$.

4.2 Overall Methodology

In our proposed approach, we eliminate the dynamic feature points when estimating the mobile robot odometry, in order to improve the mobile robot odometry accuracy by reducing the impact of moving objects. We estimate the fundamental matrix F' using the IMU motion model to classify the matched feature points as static or dynamic. Thereafter, the dynamic feature points are eliminated, and the refined feature points are used for visual odometry (VO). Finally, VO is fused with inertial and wheel encoder data to produce accurate odometry results. The VO front-end utilizes oriented fast and rotated brief (ORB)

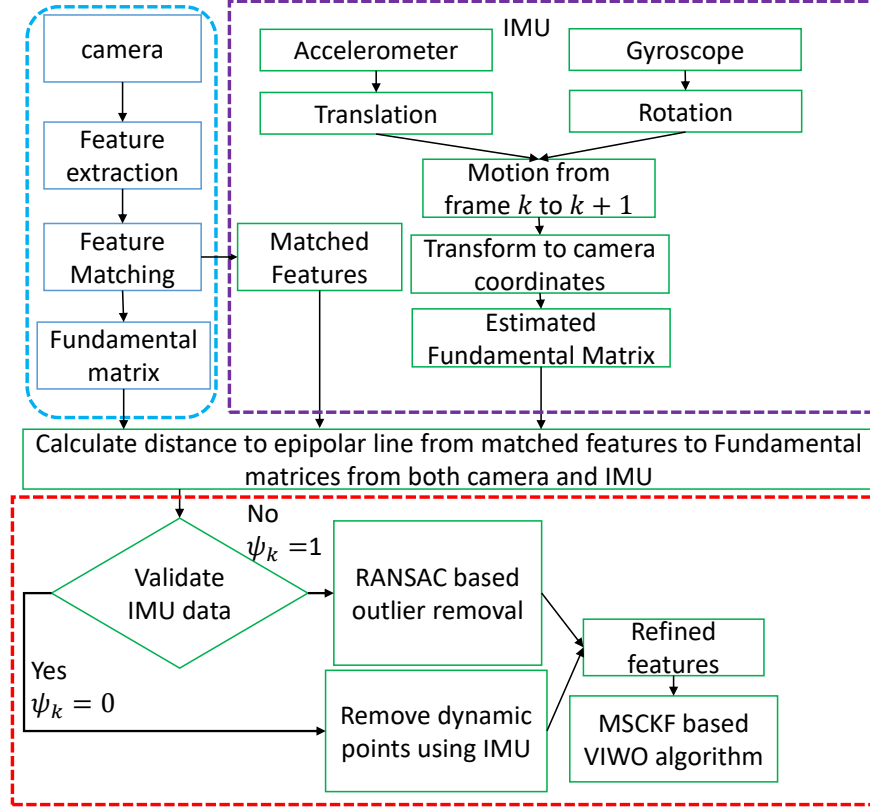


Figure 4.1: Proposed robot pose estimation method with dynamic feature point removal

[102] features extracted from subsequent images to estimate the robot motion from visual information. The proposed robot pose estimation scheme is shown in Fig. 4.1.

The linear velocity $\mathbf{v}$ of the current IMU frame with respect to the global frame of the robot is obtained by integrating the measured acceleration $\mathbf{a}_m$ by the IMU given as

$${}^G\mathbf{v}_{I_{k+1}} = {}^G\mathbf{v}_I(t_k) \quad (4.25)$$

where

$${}^G\mathbf{v}_I(t) = {}^G\mathbf{v}_{I_k} + \int_{\tau=t_k}^t ({}^G\mathbf{C}_I(\tau)(\mathbf{a}_m - \mathbf{b}_a) - {}^G\mathbf{g}) d\tau$$

where ${}^G\mathbf{C}_I(t)$ is the orientation of the robot represented by the rotation matrix in the global frame at time $t \in [t_k, t_{k+1}]$. Further integrating (4.25), the translation of the system

between two inertial frames I_k and I_{k+1} with respect to the global frame is obtained as

$${}^G\mathbf{p}_{I_{k+1}} = {}^G\mathbf{p}_{I_k} + \int_{t_k}^{t_{k+1}} ({}^G\mathbf{v}_I(\tau)) d\tau \quad (4.26)$$

Similarly, the IMU also measures the angular velocity $\boldsymbol{\omega}$ of the robot. The rotation $\mathbf{q}$ of the robot between two frames is obtained by integrating the angular velocity given as

$${}^G\mathbf{q}_{I_{k+1}} = {}^G\mathbf{q}_{I_k} \otimes \int_{\tau=t_k}^{t_{k+1}} \left(\frac{1}{2} \boldsymbol{\Omega}(\boldsymbol{\omega}_m - \mathbf{b}_g) {}^I\mathbf{q}_G \right) d\tau. \quad (4.27)$$

The obtained translation and rotation motions are represented in the camera coordinate system, utilizing the camera-IMU extrinsic parameters, since the camera and IMU have a rigid connection. Hence, the fundamental matrix $\mathbf{F}'$ is estimated with IMU data utilizing the camera-IMU extrinsics. This estimated fundamental matrix $\mathbf{F}'$ is then used to eliminate dynamic feature points, as detailed in Section 4.4.

4.3 MSCKF Based Robot Odometry Estimation

Propagation of the inertial state $\mathbf{x}_I$ is represented by the generic IMU nonlinear kinematics process model [22]

$$\mathbf{x}_{I_{k+1}} = \mathbf{f}(\mathbf{x}_{I_k}, \mathbf{a}_m, \boldsymbol{\omega}_m, \mathbf{n}_I), \quad (4.28)$$

where $\mathbf{n}_I$ denotes zero-mean white Gaussian IMU measurement noise. The state estimate is generated as

$$\hat{\mathbf{x}}_{I_{k+1}} = \mathbf{f}(\hat{\mathbf{x}}_{I_k}, \mathbf{a}_m, \boldsymbol{\omega}_m, \mathbf{0}). \quad (4.29)$$

The corresponding state covariance matrix is propagated via

$$\mathbf{P}_{k+1} = \boldsymbol{\Phi}_k \mathbf{P}_k \boldsymbol{\Phi}_k^\top + \mathbf{Q}_k, \quad (4.30)$$

where $\boldsymbol{\Phi}_k$, $\mathbf{Q}_k$ are the system Jacobian and noise covariance matrices [44]. Once the camera and WO measurement functions are obtained using the measurement models given in (4.8),(4.9), the robot pose $\mathbf{x}$ is estimated using an extended Kalman filter (EKF), as follows. The bearing measurement model (4.5) is rewritten in the form

$$\mathbf{z}_{m,k} = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}_{m,k} \quad (4.31)$$

where the measurement noise is assumed to be $\mathbf{n}_{m,k} \sim \mathcal{N}(0, \mathbf{R}_{m,k})$. (4.31) is linearized as,

$$\tilde{\mathbf{z}}_{m,k} = \mathbf{H}_k \tilde{\mathbf{x}}_k + \mathbf{n}_{m,k} \quad (4.32)$$

where $\mathbf{H}_k$ is the measurement Jacobian. Using this linearized measurement model, the EKF update equations are set as,

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \mathbf{K}_k \tilde{\mathbf{z}}_{m,k}, \quad (4.33)$$

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_k, \quad (4.34)$$

$$\mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^\top (\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^\top + \mathbf{R}_{m,k+1})^{-1}, \quad (4.35)$$

where $\mathbf{K}_k$ is the Kalman gain.

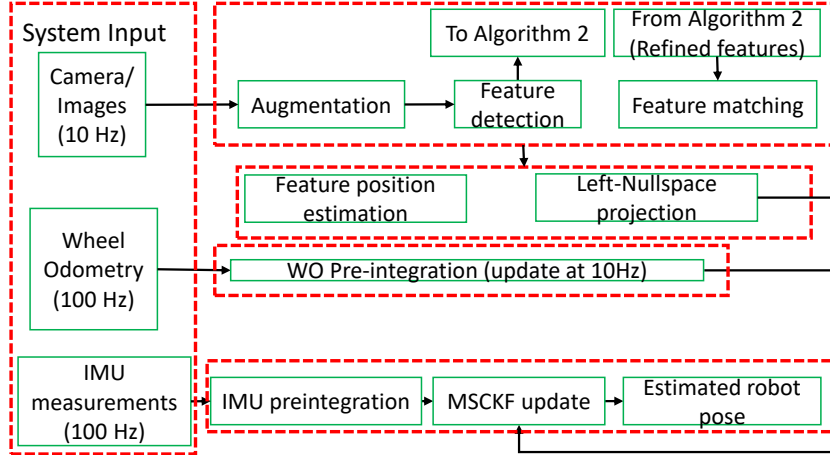


Figure 4.2: MSCKF framework for robot pose estimation

4.4 IMU Based Confidence Estimation and Outlier Removal

The dynamic point elimination method utilizing IMU data is summarized in Algorithm 2. The fundamental matrix $\mathbf{F}$ is estimated using image data. The distance between the corresponding feature points and the epipolar line is calculated for both $\mathbf{F}$ and $\mathbf{F}'$ as $d_{k,i}^1$ and $d_{k,i}^2$. Comparing the two distances $d_{k,i}^1$ and $d_{k,i}^2$ to a threshold value ρ , we determine whether each matched feature point is static or dynamic via

$$\phi_{k,i} = \begin{cases} 1, & |d_{k,i}^1 - d_{k,i}^2| \geq \rho \\ 0, & |d_{k,i}^1 - d_{k,i}^2| < \rho \end{cases} . \quad (4.36)$$

The value $\phi_{k,i}$ is set to 0 if the i^{th} feature point of the k^{th} frame is consistent with the fundamental matrix. Similarly, it is set to 1 if otherwise. Once the dynamic points are eliminated, we get reliable VO and is fused with inertial and WO data in an MSCKF framework as shown in Fig. 4.2.

Algorithm 2: Algorithm for dynamic point elimination

Input : matched feature points of adjacent frames

$$\Pi_k = \{\mathbf{p}_{k,1}, \dots, \mathbf{p}_{k,n}\}, \Pi_{k+1} = \{\mathbf{p}_{k+1,1}, \dots, \mathbf{p}_{k+1,n}\}$$

Output: refined feature points with dynamic feature points removed

while $t \leq t_{\text{image}}$ **do**

for each point $\mathbf{p}_{k,i} \in \Pi_k$ **and** $\mathbf{p}_{k+1,i} \in \Pi_{k+1}$ **do**

$$[x, y, z]^T = \mathbf{F} \mathbf{p}_{k,i} ;$$

$$d_{k,i}^1 = \frac{|\mathbf{p}_{k+1,i}^T \mathbf{F} \mathbf{p}_{k,i}|}{\sqrt{\|x\|^2 + \|y\|^2}} ;$$

$$[x', y', z']^T = \mathbf{F}' \mathbf{p}_{k+1,i}$$

$$d_{k,i}^2 = \frac{|\mathbf{p}_{k,i}^T \mathbf{F}' \mathbf{p}_{k+1,i}|}{\sqrt{\|x'\|^2 + \|y'\|^2}} ;$$

if $\psi_k = 1$ **and** $|d_{k,i}^1 - d_{k,i}^2| \geq \rho$ **then** remove dynamic points $\mathbf{p}_{k,i}, \mathbf{p}_{k+1,i}$ from Π_k, Π_{k+1} ;

else if $\psi_k = 0$ **then**

 Remove outliers according to RANSAC with threshold ζ as given in

 Eq.(4.40)

end

end

end

Before using the IMU data to eliminate dynamic points we need to validate them. To check this, we use another threshold μ where the valid feature match count is given as

$$\psi_k = \begin{cases} 0, & \sum_{i=1}^m \phi_{k,i} < \mu \\ 1, & \sum_{i=1}^m \phi_{k,i} \geq \mu \end{cases} . \quad (4.37)$$

If the IMU data is valid (i.e $\psi_k = 0$) and IMU data is trustworthy and can be used in dynamic point elimination. Otherwise, the system switches to RANSAC-based outlier

removal scheme based on trifocal geometry, as indicated in Fig. 4.1. The trifocal tensor relates the projective geometric relationship among three subsequent views of the camera, as indicated in Fig. 4.3. This method is used as an alternative to our proposed approach in the case that the confidence in IMU-based dynamic point elimination is low.

Figure 4.3 illustrates image coordinates m_1, m_2, m_3 of the three views obtained by the viewpoints C_1, C_2, C_3 . For m_1 and m_2 the fundamental matrix $\mathbf{F}$ satisfies the epipolar geometry condition $\bar{\mathbf{m}}_2^\top \mathbf{F} \bar{\mathbf{m}}_1 = 0$, where $\bar{\mathbf{m}}_i = \mathbf{K}^{-1} \mathbf{m}_i$ denotes the normalized image coordinates given the camera intrinsic matrix $\mathbf{K}$. For the three-view case, the geometry constraints are given such that the line l_2 is perpendicular to the epipolar line in the second view. Given the camera projection matrices $\mathbf{P}_1 = [\mathbf{I}|\mathbf{0}]$, $\mathbf{P}_2 = [\mathbf{A}|\mathbf{a}_4]$, $\mathbf{P}_3 = [\mathbf{B}|\mathbf{b}_4]$, where $\mathbf{I}, \mathbf{A}, \mathbf{B}$ are the rotation matrices and $\mathbf{0}, \mathbf{a}_4, \mathbf{b}_4$ are the translation vectors, the 3-by-3-by-3 trifocal tensor matrix $\mathbf{T} = [\mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3]$ is given by

$$\mathbf{T}_i = \mathbf{b}_4 \mathbf{a}_i^\top - \mathbf{a}_4 \mathbf{b}_i^\top, \quad (4.38)$$

where $\mathbf{a}_i$, and $\mathbf{b}_i$ are the i^{th} column elements of $\mathbf{P}_2, \mathbf{P}_3$. Utilizing this trifocal tensor the point-line-point correspondence illustrated in Figure 4.3, the corresponding geometric constraint

$$\bar{\mathbf{m}}_3 = \left(\sum_{i=1}^3 \bar{\mathbf{m}}_{1,i} \mathbf{T}_i^\top \right) l_2$$

is used to construct the measurement model

$$\mathbf{z}_T = \begin{bmatrix} \bar{\mathbf{m}}_2^\top \mathbf{R}_{12}^\top \langle \mathbf{t}_{12} \times \rangle \bar{\mathbf{m}}_1 \\ \bar{\mathbf{m}}_3^\top \mathbf{R}_{23}^\top \langle \mathbf{t}_{23} \times \rangle \bar{\mathbf{m}}_2 \\ \mathbf{K} \left(\sum_{i=1}^3 \bar{\mathbf{m}}_{1,i} \mathbf{T}_i^\top \right) l_2 \end{bmatrix}, \quad (4.39)$$

where $\mathbf{R}_{ij}$ denotes the rotation matrix from i to j . $\langle \mathbf{t}_{ij} \times \rangle$ is the skew-symmetric matrix of relevant translation vector.

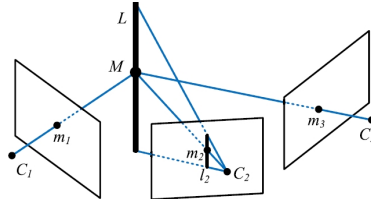


Figure 4.3: Three view geometry

Here, a user-defined threshold value ζ is used for outlier removal using [RANSAC](#), and the acceptable (inlier) set M for $(\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3)$ after refinement by [RANSAC](#) is set as [\[55\]](#)

$$M = \left\{ (\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3) \mid \left\| \mathbf{m}_3 - \mathbf{K} \left(\sum_{i=1}^3 \bar{\mathbf{m}}_{1i}^1 \mathbf{T}_i^\top \right) \right\|_2 < \zeta \right\}. \quad (4.40)$$

4.5 Real-world Data Based Simulations on Dynamic Point Elimination

4.5.1 Dynamic Point Elimination

The proposed scheme is verified via real-time data simulations based on an experimental [\[58\]](#) dataset which is collected in an urban area with dynamic objects. The sensor specifications of the dataset are given in [Table 5.1](#). To show the effectiveness of the proposed scheme, we compare the odometry results obtained in two cases. Firstly, we compare the odometry results obtained without fusing WO for two cases with and without dynamic points elimination. Secondly, we do the same comparison including WO. This process is repeated for a total of 5 trajectory sequences to further verify the proposed method. As seen in [Figs. 4.4\(a\)](#), and [4.5\(a\)](#), the results indicated by the red line, with dynamic features elimination outperforms the results that are indicated by a violet line, which is without dynamic point elimination. We do not consider loop closure, as our objective is to examine the odometry estimation accuracy and its improvement. Specifically, for the first robot trajectory in the case with WO shown by the dashed red line the [RMSE](#) was 4.350 m as opposed to when without dynamic point elimination shown by the dashed blue line where the [RMSE](#) was 4.913 m. Similar improvements could be seen in the second robot trajectory, where the [RMSE](#) was increased from 3.214 m to 1.756 m. As shown in [Table 4.1](#) the average root mean squared error (ARMSE) per km is calculated for each scenario. The ARMSE was lower for each scenario when we consider dynamic point elimination. Furthermore, as indicated in the [Figs. 4.4\(b\)](#), and [4.5\(b\)](#), the fusion of WO improves the overall accuracy of the robot trajectory estimation, making it comparable with the ground truth trajectory.

To examine the estimated trajectory, we also embed the translation and rotation error plots with its respective 3-sigma error bands, shown by red dashed lines. [Figure 4.6a](#) provides the translation error propagation across time for trajectory 1 shown by the cyan solid line. As we can see from the plot the 3 sigma gap widens indicating the inherent drift

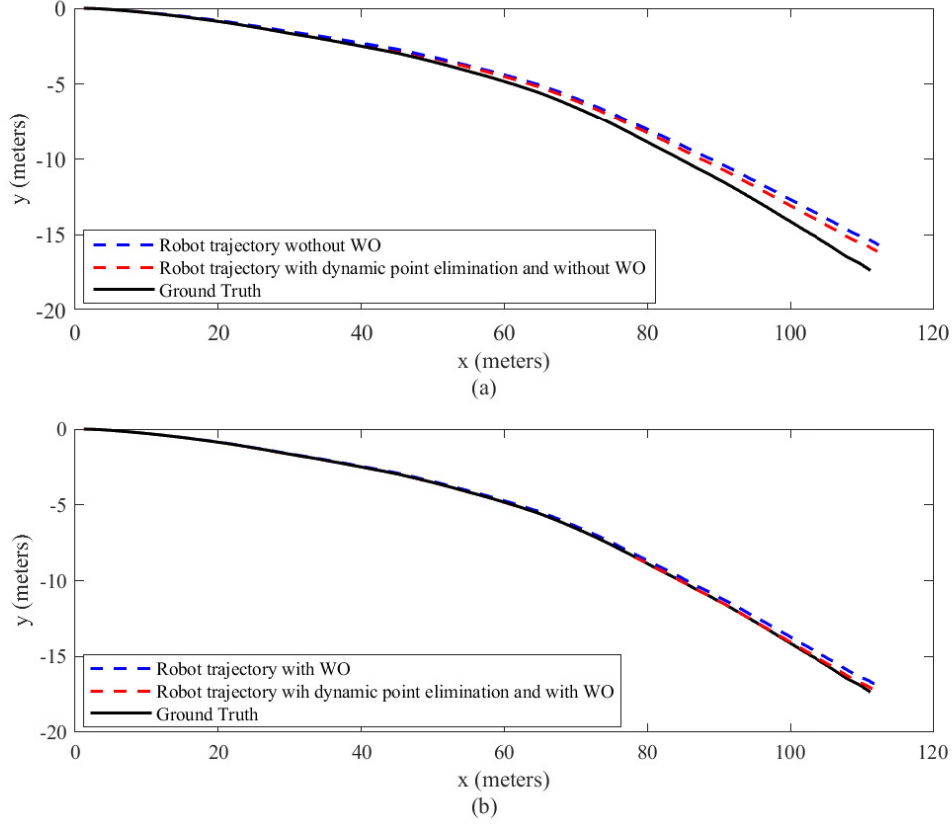


Figure 4.4: Comparison of robot trajectory 1 with and without dynamic point elimination for the cases of (a) VIO (b) VIWO

issue of odometry. Figure 4.6b outlines consistent error in relation to rotation where the trajectory closely follows the reference trajectory. Figures 4.7a and 4.7b indicate that the translation and rotation errors remains fairly constant until the sharp turn affecting the 3-sigma plots to widen. Hence, this illustrates the importance of addition of WO information to the filter as an additional corrective measure.

4.6 Summary and Conclusion

In this paper, for mobile robots navigating under heavy dynamic environments, we developed a dynamic feature point elimination methodology using IMU data and an MSCKF-based system to estimate robot position and orientation. The proposed dynamic feature

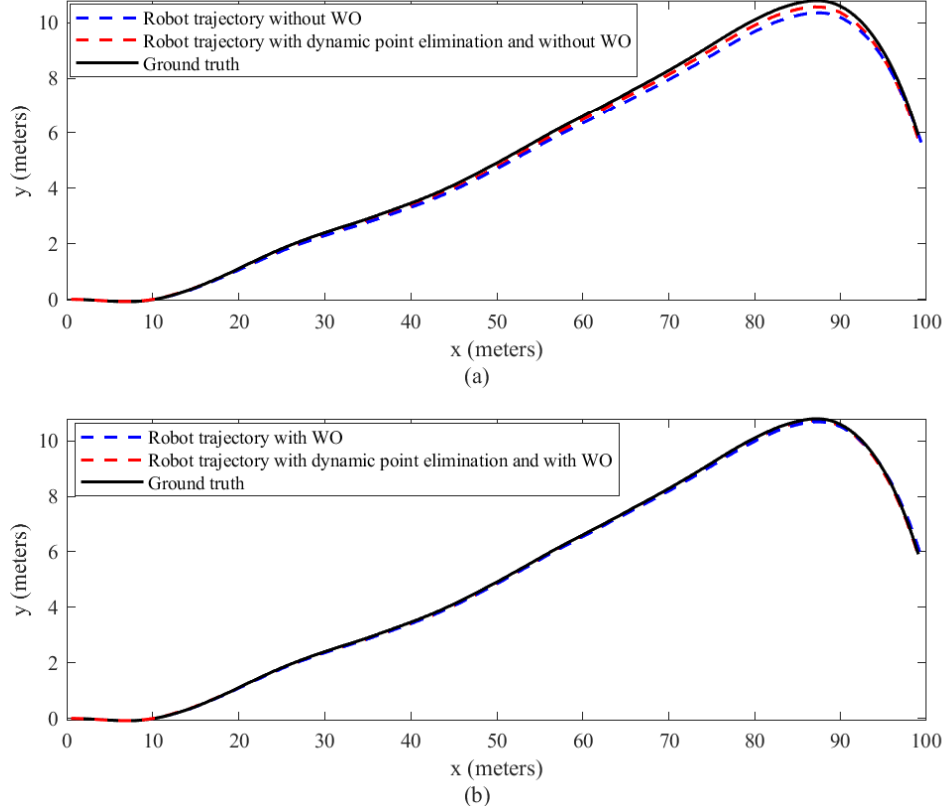
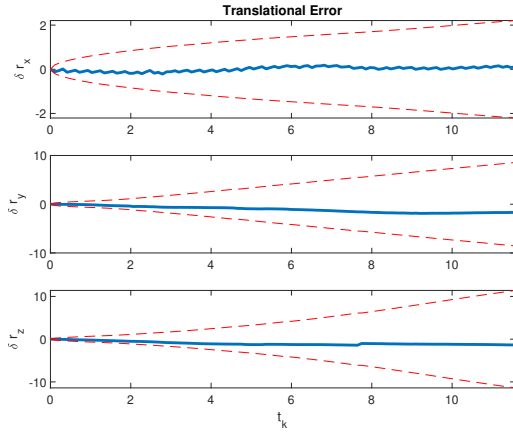
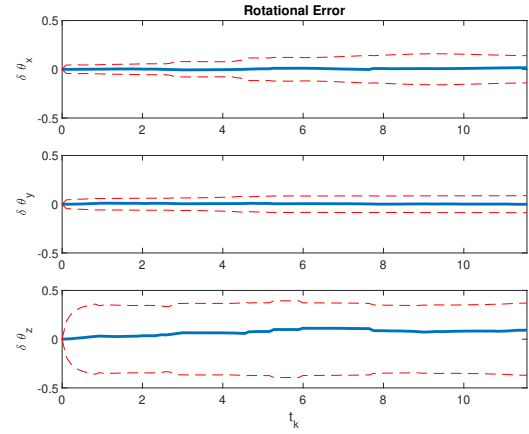


Figure 4.5: Comparison of robot trajectory 2 with and without dynamic point elimination for the cases of (a) VIO (b) VIWO.

point elimination methodology is based on a geometric feature matching confidence checking procedure. When the confidence in the IMU-based estimator is low, the proposed system design switches back to [RANSAC](#)-based outlier removal. The conducted real-data-based simulation tests demonstrate that the proposed methodology increases the accuracy of the robot odometry. Furthermore, we extend the devised [Algorithm 2](#) in by incorporating Semantic Segmentation to remove dynamic feature points of detected dynamic objects, which is presented in [Chapter 6](#). One future research direction of this project could be the integration of [IMU](#) and [WO](#) data to further enhance the accuracy of the dynamic feature point elimination.

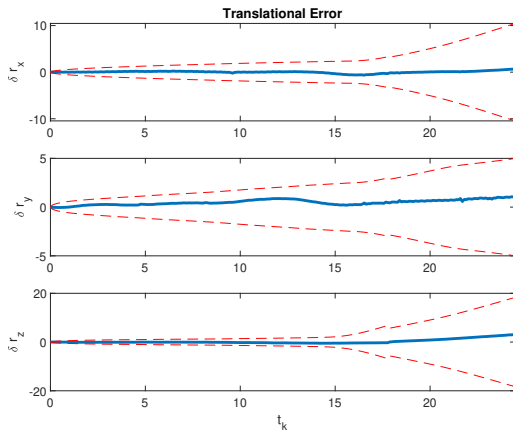


(a) Translation error.

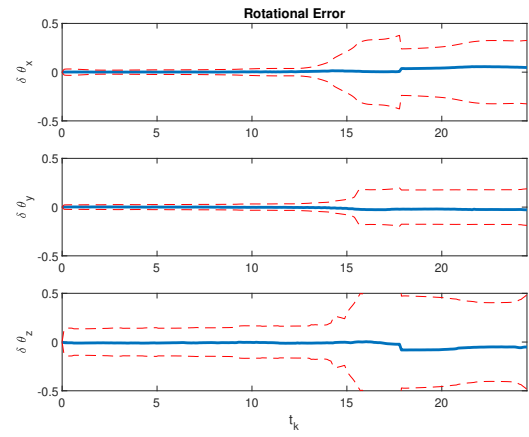


(b) Rotation error.

Figure 4.6: Trajectory 1 translation and rotation error with 3-sigma error bounds.



(a) Translation error.



(b) Rotation error.

Figure 4.7: Trajectory 2 translation and rotation error with 3-sigma error bounds.

Translation error (RMSE) in meters			
Without WO		With WO	
With dynamic point elimination	without dynamic point elimination	With dynamic point elimination	without dynamic point elimination
4.767	5.321	4.350	4.913
2.253	3.714	1.756	3.214
6.225	7.002	5.549	6.319
4.140	5.837	3.601	5.291
4.537	5.255	4.315	5.032

Table 4.1: Translation error with and without dynamic point elimination.

Chapter 5

Integration of Wheel Slip Compensation to VINS

The fusion of monocular visual, inertial, and wheel encoder measurements has recently gained immense interest as a robot odometry and localization approach that overcomes the effects of navigation system uncertainties and variations. However, the wheel encoder measurements fused to the visual inertial wheel odometry (VIWO) system in this approach can be faulty, mainly due to wheel slippages and other inherent errors. This chapter proposes a strategy for compensating wheel slip effects, based on a differential drive robot kinematics model. We use Gaussian processes to learn the error between the WO model and the ground-truth for a set of training sequences. The learned WO error information is then used on the test sequences to correct the errors in WO. Then, the corrected WO measurements are utilized in a MSCKF based state estimation scheme to obtain localization information. The enhancement is demonstrated via real-world data simulations.

5.1 Problem and Approach

VISLAM algorithms are considered state of the art for localization where GPS is not available. The initialization procedure of VINS algorithms plays an important role in the overall accuracy and robustness of the algorithm. Visual-inertial initialization can be classified as joint or disjoint initialization estimation procedures [17]. The joint estimation procedures try to build a closed-form algebraic system of equations and solve using linear/non-linear least squares. In addition, the disjoint estimation methods are made on top of the assumption that up-to-scale camera trajectory can be obtained accurately using monocular

vision, thereby using this trajectory to estimate the inertial parameters. Regardless of these methods, as described in section 2.4 there are certain degenerate motions that a wheeled mobile robot undergoes and fails to initialize or initialize with very large errors affecting their overall accuracy and robustness. To tackle this issue, researchers have identified integrating wheel encoder measurement information can be used to negate the initialization errors. However, WO measurements are greatly affected by slip. Hence, considering these challenges this chapter discusses a methodology using visual, inertial, and vehicle dynamics measurements based on a high-fidelity differential drive motion dynamics model to estimate and the state of the robot by compensating for the wheel slippage which adversely affect the accuracy of localization. This is a key issue that has not gained attention, which can improve VINS algorithms through aided WO.

Section 2.4 provided the challenges and background information on related material. Section 5.2 summarizes the key measurement models used to estimate the wheel slip in order to improve the accuracy of the whole system.

5.2 Wheel Slippage Estimation

This section outlines the measurement information used to estimate the odometry.

5.2.1 Inertial Measurement Model

The inertial state $\bar{\mathbf{x}}_{I_k}$ is measured using a 6-axis IMU continuously. The local angular velocity and linear acceleration measurements by the IMU at each step k are given by [20]

$$\omega_{m,k} = \omega_k + \mathbf{b}_g + \mathbf{n}_g, \quad (5.1)$$

$$\mathbf{a}_{m,k} = \mathbf{a}_k + {}^b\mathbf{R}_g {}^w\mathbf{g} + \mathbf{b}_a + \mathbf{n}_a, \quad (5.2)$$

where ω and $\mathbf{a}$, respectively, denote the actual angular velocity and acceleration. $\mathbf{n}_g$ and $\mathbf{n}_a$ denote the gyroscope and accelerometer measurement noises that are assumed to be zero-mean Gaussian random processes. ${}^w\mathbf{g} \approx [0 \ 0 \ 9.81]^T$ is the global gravity vector. The camera measurements are obtained at a sampling time $\Delta \bar{t} = 0.1$ sec, which is 10 times the main (inertial) sampling time Δt . We denote this factor as $\tau = \frac{\Delta \bar{t}}{\Delta t} = 10$. Accordingly, the

inertial state vector at the camera and odometry measurement sampling time $t = \bar{t}_k = \bar{k}\Delta\bar{t}$ is defined as

$$\mathbf{x}_{I_{\bar{k}}} = [\mathbf{x}_{\bar{k}}^T \quad {}^w\mathbf{v}_{b_{\bar{k}}}^T \quad \mathbf{b}_g^T \quad \mathbf{b}_a^T]^T = \bar{\mathbf{x}}_{I_{10\bar{k}}} = [\bar{\mathbf{x}}_{10\bar{k}}^T \quad {}^w\bar{\mathbf{v}}_{b_{10\bar{k}}}^T \quad \mathbf{b}_g^T \quad \mathbf{b}_a^T]^T. \quad (5.3)$$

Below we define the pre-integrated IMU measurements between two camera frames $C_{\bar{k}}$ and $C_{\bar{k}+1}$. The pre-integrated velocity ${}^w\mathbf{v}_b$ of frame $\{b\}$ with respect to frame $\{w\}$ of the robot is obtained by integrating the acceleration $\mathbf{a}_m$ measured by the IMU given as

$${}^w\mathbf{v}_{b_{\bar{k}+1}} = {}^w\mathbf{v}_{b_{\bar{k}}} + \Delta\bar{t} \sum_{q=10\bar{k}}^{10\bar{k}+10} ({}^w\mathbf{R}\mathbf{a}_{m,q} - {}^w\mathbf{g}), \quad (5.4)$$

where ${}^w\mathbf{R}$ is the orientation of the robot represented by the rotation matrix in world frame. Further integrating (5.4), the translation of the system between two inertial frames $b_{\bar{k}}$ and $b_{\bar{k}+1}$ with respect to the world frame is obtained as

$${}^w\mathbf{p}_{b_{\bar{k}+1}} = {}^w\mathbf{p}_{b_{\bar{k}}} + {}^w\mathbf{v}_{b_{\bar{k}}} \Delta\bar{t} + \Delta\bar{t}^2 \sum_{q=10\bar{k}}^{10\bar{k}+10} \sum_{p=10\bar{k}}^q ({}^w\mathbf{R}\mathbf{a}_{m,p} - {}^w\mathbf{g}). \quad (5.5)$$

Similarly, the IMU also measures the angular velocity ω of the robot. The rotation $\mathbf{q}$ of the robot between two frames is obtained by integrating the angular velocity given as

$${}^w\mathbf{q}_{b_{\bar{k}+1}} = {}^w\mathbf{q}_{b_{\bar{k}}} \otimes \Delta\bar{t} \sum_{q=10\bar{k}}^{10\bar{k}+10} \left(\frac{1}{2} \boldsymbol{\Omega}(\omega_{m,q}) {}^b\mathbf{q}_q \right). \quad (5.6)$$

Using the pre-integrated measurements (5.4),(5.5),(5.6), the inertial dynamics is represented in terms of the state $\mathbf{x}_I$ given in (5.3) by the generic process model [22]

$$\mathbf{x}_{I_{\bar{k}+1}} = f(\mathbf{x}_{I_{\bar{k}}}, \mathbf{a}_{m,\bar{k}}, \omega_{m,\bar{k}}, \mathbf{n}_{I,\bar{k}}), \quad (5.7)$$

the IMU measurement noise is represented by the zero-mean Gaussian random process $\mathbf{n}_I = [\mathbf{n}_a^T, \mathbf{n}_g^T]^T$.

This section provides details into the wheel slip estimation scheme utilized to improve the accuracy of the wheeled mobile robot dynamic measurement before pre-integrating with visual-inertial measurements. For the camera measurement function and derivation, we refer the reader back to the derivation of Eq.(4.8) in Chapter 4.

5.2.2 Wheel Encoder Measurements

The wheel encoder (i.e., WO) measurements of the robot orientation and position are represented in frame $\{o\}$ by $\mathbf{x}_{o_k} = [\mathbf{o}\mathbf{q}_{G,k}^\top \ (\mathbf{G}\mathbf{p}_{O_k}^\mathbf{G})^\top]^\top$, where $\mathbf{o}\mathbf{q}_{G,k}$ is the quaternion corresponding to the rotation $\mathbf{G}\mathbf{R}_O$. The corresponding ordinary differential equation for robot motion is expressed as

$$\begin{aligned}\mathbf{G}\dot{\mathbf{R}}_O &= \mathbf{G}\mathbf{R}_O[\mathbf{o}\boldsymbol{\omega}]_\times, \\ \mathbf{G}\dot{\mathbf{p}}_O^\mathbf{G} &= \mathbf{G}\dot{\mathbf{v}}_O^\mathbf{G} = \mathbf{G}\mathbf{R}_O \mathbf{o}\mathbf{v},\end{aligned}\tag{5.8}$$

where $\mathbf{o}\boldsymbol{\omega}$ is the angular velocity in frame $\{O\}$ which is denoted as $\mathbf{o}\boldsymbol{\omega} = \begin{bmatrix} 0 & 0 & \frac{v_r - v_l}{b} \end{bmatrix}^\top$, $\mathbf{o}\mathbf{v} = \begin{bmatrix} \frac{v_l + v_r}{2} & 0 & 0 \end{bmatrix}^\top$ is the linear velocity in frame $\{o\}$, v_l, v_r are the left and right wheel velocities, respectively. The wheel encoder provides measurements at a significantly higher frequency (100 Hz) as opposed to the camera (10 Hz). Therefore, it is computationally too costly to implement the MSCKF update at the rate of wheel encoder measurements. Hence, they are pre-integrated between two consecutive imaging time steps k and $k + 1$ [158]. Considering this pre-integration, the WO measurement model is formed as,

$$\mathbf{z}_{o_{\bar{k}+1}} = \int_{\tau=t_k}^{t_{k+1}} \begin{bmatrix} \frac{v_{r,k} - v_{l,k}}{b} \\ \frac{v_{l,k} + v_{r,k}}{2} \cos(\theta_k) \\ \frac{v_{l,k} + v_{r,k}}{2} \sin(\theta_k) \end{bmatrix} d\tau + \mathbf{n}_{o_{\bar{k}}} = \begin{bmatrix} z_\theta \\ z_x \\ z_y \end{bmatrix} + \mathbf{n}_{o_{\bar{k}}},\tag{5.9}$$

where $\mathbf{n}_{o_{\bar{k}}}$ denotes the 3×1 zero-mean Gaussian measurement noise vector. Considering the approximate planar motion (in $x - y$ plane) of the robot, we augment the measurement model as

$$\bar{\mathbf{z}}_{o_{\bar{k}+1}} = \begin{bmatrix} z_x \\ z_y \\ 0 \\ 0 \\ 0 \\ z_\theta \end{bmatrix} + \bar{\mathbf{n}}_{o_{\bar{k}}} = \mathbf{h}(\mathbf{x}_{o_{\bar{k}}}, \mathbf{o}\mathbf{v}_k, \mathbf{o}\boldsymbol{\omega}_k) + \bar{\mathbf{n}}_{o_{\bar{k}}},\tag{5.10}$$

where $\bar{\mathbf{n}}_{o_{\bar{k}}}$ is the augmented 6×1 noise vector having $\mathbf{n}_{o_{\bar{k}}}$ entries in its first, second, and last entries, and having the other three entries equal to zero.

5.3 Overall Methodology

It is a clear issue that **WO** suffers from accumulative errors due to wheel slippage when undergoing motion in uneven or slippery terrains [97]. As a solution to this **VO** is used for mobile robots and generally they perform well. However, as discussed in section 1.1 **VO** has its shortcomings. These issues are compensated using **VIO** algorithms. Furthermore, **VIO** algorithms can be pre-integrated with wheel odometry information to increase the localization accuracy. The accuracy of this **WO** pre-integration depends on whether the wheels undergo heavy slippage or not. Hence, to improve the accuracy of **VIWO** wheel slippage compensation is proposed for algorithms that have to undergo motion in uneven or slippery terrains. In addition, **VO** is not suitable for this purpose as it has a very low sampling rate compared to other sensors such as **IMU** and wheel encoders [81].

The robot considered in this chapter is equipped with wheel encoders that measure the left and right wheel linear velocities v_l, v_r , an accurate fiber optics gyro (**FOG**) which provides accurate heading velocity $\dot{\psi}$, an **IMU** and a stereo camera. The dynamical model used in our application is a differential drive robot (see Figure 5.1).

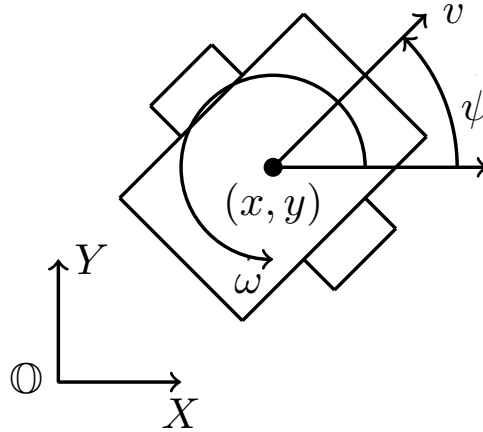


Figure 5.1: Robot model.

The outline of our proposed **VIWO** scheme is shown in Figure 5.2. We take in three streams of data for our **VIWO** state estimation. The camera provides data that is used to obtain front-end **VO**. **IMU** data are pre-integrated to match with camera data. Finally, we utilize **WO** data which is compensated for slippage and other errors before pre-integration. Once the slippage is compensated, the **WO** data are pre-integrated to match with camera

data. Then, the three data streams are fused together in an MSCKF framework to produce **VIWO** state estimates.

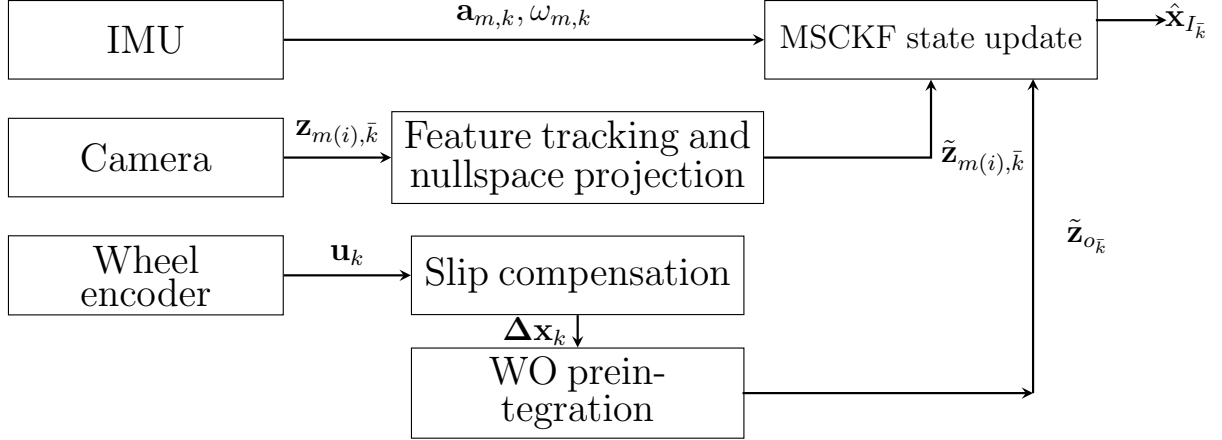


Figure 5.2: Outline of the proposed VIWO scheme.

The slip compensation methodology is described in Figure 5.3 and is explained in detail in Section 5.4.

5.4 Slip Compensation via Gaussian Process Regression

Our slip compensator design involves learning-based estimation of the (error) residuals between ground truth and physical **WO** model considering a planar motion differential drive robot. In training this learning-based estimator design utilizing Gaussian process regression (**GPR**), the robot odometry dynamics is represented by the state vector

$$\xi_k = \begin{bmatrix} \xi_{1k} \\ \xi_{2k} \\ \xi_{3k} \end{bmatrix} = [x_k \quad y_k \quad \psi_k]^\top, \quad (5.11)$$

where (x_k, y_k) denotes the position of the robot and ψ_k denotes the orientation of the robot at training sample k for the residual estimator design in the sequel. The inputs of the dynamic system are the velocity v_k and the heading angle increment $\delta\psi_k$ (measured by the **FOG**), represented in stacked vector form by

$$\mathbf{u}_k = [v_k \quad \delta\psi_k]^\top, \quad (5.12)$$

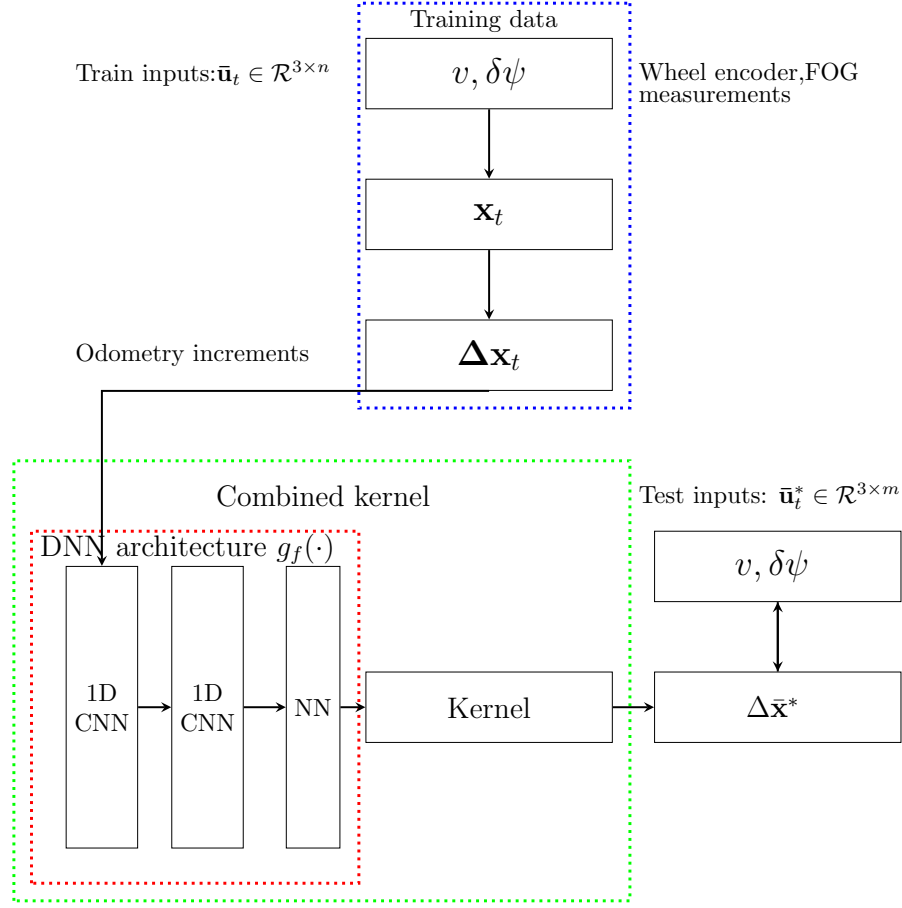


Figure 5.3: Slip compensation scheme.

where $v = \frac{v_l + v_r}{2}$, v_l is the left wheel speed, and v_r is the right wheel speed. For representing the state increment $\Delta \xi_k$ with input $\mathbf{u}_k$, we use the discrete-time dynamic system model [19]

$$\begin{aligned} \Delta \xi_k &= \xi_{k+1} - \xi_k = \bar{\mathbf{u}}_k + \mathbf{w}_k, \\ \bar{\mathbf{u}}_k &= \begin{bmatrix} \bar{u}_{1k} \\ \bar{u}_{2k} \\ \bar{u}_{3k} \end{bmatrix} = \mathbf{f}_o(\mathbf{u}_k) = \begin{bmatrix} v_k \cos(\delta\psi_k) \delta_k \\ v_k \sin(\delta\psi_k) \delta_k \\ \delta\psi_k \end{bmatrix}, \quad \mathbf{w}_k = \begin{bmatrix} w_{1k} \\ w_{2k} \\ w_{3k} \end{bmatrix}, \end{aligned} \quad (5.13)$$

where $\mathbf{w}_k$ denotes the Gaussian process noise such that $w_{ik} \sim \mathcal{N}(0, \sigma_{w_{ik}}^2)$ and Δk denotes the sampling time. Next, a GPR is formulated for predictive correction of the odometry

increments $\Delta\xi_k$ governed by (5.13). Consider a sequence of inputs (from training data sequences) collected in the individual input vector $\bar{\mathbf{U}}_i = [u_{i_k}, \dots, u_{i_{kn}}]^\top \in \mathcal{R}^n$ and the corresponding sequence of state increments collected in the vector $\Delta\mathbf{X}_i = [\Delta\xi_{i_k}, \dots, \Delta\xi_{i_{kn}}]^\top \in \mathcal{R}^n$. For predictions $\Delta\mathbf{X}_i^* = [\Delta\xi_{i_k}^*, \dots, \Delta\xi_{i_{km}}^*]^\top$ on new data $\bar{\mathbf{U}}_i^* \in \mathcal{R}^m$ (test sequences), the predictive inference is given by

$$p(\Delta\mathbf{X}_i^* | \bar{\mathbf{U}}_i, \Delta\mathbf{X}_i, \bar{\mathbf{U}}_i^*) \sim \mathcal{N}(\Delta\bar{\mathbf{X}}_i^*, \text{cov}(\Delta\mathbf{X}_i^*)), \quad (5.14)$$

where

$$\Delta\bar{\mathbf{X}}_i^* = \mathbf{K}(\bar{\mathbf{U}}_i^*, \bar{\mathbf{U}}_i) [\mathbf{K}(\bar{\mathbf{U}}_i, \bar{\mathbf{U}}_i) + \sigma_{w_i}^2 \mathbf{I}]^{-1} \Delta\mathbf{X}_i \quad (5.15)$$

is the mean prediction,

$$\text{cov}(\Delta\bar{\mathbf{X}}_i^*) = \mathbf{K}(\bar{\mathbf{U}}_i^*, \bar{\mathbf{U}}_i^*) - \mathbf{K}(\bar{\mathbf{U}}_i^*, \bar{\mathbf{U}}_i) [\mathbf{K}(\bar{\mathbf{U}}_i, \bar{\mathbf{U}}_i) + \sigma_{w_i}^2 \mathbf{I}]^{-1} \mathbf{K}(\bar{\mathbf{U}}_i, \bar{\mathbf{U}}_i^*) \quad (5.16)$$

is the prediction covariance, $\mathbf{K}(\bar{\mathbf{U}}_i^*, \bar{\mathbf{U}}_i^*) \in \mathcal{R}^{m \times m}$, $\mathbf{K}(\bar{\mathbf{U}}_i^*, \bar{\mathbf{U}}_i) \in \mathcal{R}^{m \times n}$, $\mathbf{K}(\bar{\mathbf{U}}_i, \bar{\mathbf{U}}_i^*) \in \mathcal{R}^{n \times m}$, and $\mathbf{K}(\bar{\mathbf{U}}_i, \bar{\mathbf{U}}_i) \in \mathcal{R}^{n \times n}$ are the kernel matrices where the element (i,j) of $\mathbf{k}_{ij} = \mathbf{k}(u_i, u_j)$. $\mathbf{k}(\cdot, \cdot)$ is the kernel function which consists of the base kernel $\bar{\mathbf{k}}$ (Matern kernel [138]) combined with the NN (neural network) architecture $g_f(\cdot)$ in the form [12]

$$\mathbf{k}(\cdot, \cdot) = \bar{\mathbf{k}}(g_f(\cdot), g_f(\cdot)). \quad (5.17)$$

In (5.17), $\bar{\mathbf{k}}(u_i, u_j) = \sigma_k^2 \left(1 + \frac{\sqrt{5}|u_i - u_j|}{l} + \frac{\sqrt{5}|u_i - u_j|^2}{3l^2} \right) \exp \left(-\frac{\sqrt{5}|u_i - u_j|}{l} \right)$, where σ_k is the magnitude parameter and $l > 0$ is the scale parameter. Using the optimized hyperparameters (σ_k, l) , the kernel $\mathbf{k}(\cdot, \cdot)$, the corrected state vector increment $\Delta\xi_{i_q}$, for $i = 1, 2, 3$, and $q = 1, \dots, m$ is obtained. Once the corrected set of odometry increments $\Delta\xi_q$ are obtained for a test sequence, similar to in (5.10) we obtain the augmented measurement given as

$$\check{\mathbf{z}}_{o_{\bar{k}+1}} = \begin{bmatrix} \sum_{q=10\bar{k}}^{10\bar{k}+10} \Delta x_q \\ \sum_{q=10\bar{k}}^{10\bar{k}+10} \Delta y_q \\ 0 \\ 0 \\ 0 \\ \sum_{q=10\bar{k}}^{10\bar{k}+10} \Delta\psi_q \end{bmatrix}, \quad (5.18)$$

which is used by the MSCKF state estimator.

5.5 State Estimation

Based on the dynamic model developed (5.7) and the methodology of [101], we develop an MSCKF as follows: The main state update rule is in the form of

$$\hat{\mathbf{x}}_{I_{\bar{k}+1}} = f(\hat{\mathbf{x}}_{I_{\bar{k}}}, \mathbf{a}_{\mathbf{m},\bar{k}}, \omega_{\mathbf{m},\bar{k}}, \mathbf{0}) + \Delta_{I_{\bar{k}}}, \quad (5.19)$$

where $\Delta_{I_{\bar{k}}}$ represents the correction term [101] that is produced considering the camera measurement residual functions $\tilde{\mathbf{z}}_{m(i),\bar{k}}$, and the slip-compensated WO measurement residual $\tilde{\mathbf{z}}_{o_{\bar{k}}} = \check{\mathbf{z}}_{o_{\bar{k}}} - h(\hat{\mathbf{x}}_{o_{\bar{k}}}, {}^o v_{\bar{k}}, {}^o \omega_{\bar{k}})$, and the corresponding covariance matrices $\mathbf{P}_{\bar{k}}$ and $\mathbf{P}_{o_{\bar{k}}}$. Note that the MSCKF keeps track of n sliding window of poses $\mathbf{x}_{C_{\bar{k}}} = [\mathbf{x}_{\bar{k}-1}^\top \ \dots \ \mathbf{x}_{\bar{k}-n}^\top]^\top$ as described in Eq.(4.2), which leads to the augmented state estimate vector $\hat{\mathbf{x}}_{v_{\bar{k}}} = [\hat{\mathbf{x}}_{I_{\bar{k}}}^\top \ \hat{\mathbf{x}}_{C_{\bar{k}}}^\top]^\top = [\hat{\mathbf{x}}_{I_{\bar{k}}}^\top \ \hat{\mathbf{x}}_{\bar{k}-1}^\top \ \dots \ \hat{\mathbf{x}}_{\bar{k}-n}^\top]^\top$ with $\hat{\mathbf{x}}_{I_{\bar{k}}} \in \mathcal{R}^{15}$, $\hat{\mathbf{x}}_{C_{\bar{k}}} \in \mathcal{R}^{6n}$.

Note here that the robot state estimate is updated via two different types of measurements; camera and slip-compensated WO. Hence, we update the robot state estimate at each time step based on the most confident measurement using a gating test based on Mahalanobis distance γ [79]. For the visual measurement update step, we denote this distance measure as $\gamma_{v_{\bar{k}}}$. Similarly, we can obtain $\gamma_{o_{\bar{k}}}$ for wheel encoder measurement update. Then, the measurement that produces the lowest Mahalanobis distance measure (γ) is used for updating the state estimate.

5.6 Real-World Data Based Simulation on Slip Compensation

In this section, we provide experimental results obtained through publicly available datasets. We demonstrate the approach using the KAIST [58] dataset, which is collected in an urban area. KAIST dataset is chosen since it also provide wheel encoder data, which is missing in most of the available visual inertial datasets. Table 5.1 provides the KAIST dataset sensor specifications. The real-world data simulations were carried out on an Intel Core i7 laptop equipped with a NVIDIA GeForce GTX 1660 Ti GPU.

5.6.1 Dataset Training

We separate the data set into training, cross-validation, and testing. In total, 17 sequences are used for training, 4 sequences for cross-validation, and 2 sequences for testing. Table

Table 5.1: Sensor specifications for KAIST dataset.

Sensor	Manufacturer	Model	Description	freq.(Hz)	Accuracy
IMU	Xsens	MTi-300	Enhanced attitude and heading reference system	100	$10^0/h$
Camera	Pointgray	Flea3	1600×1200 RGB, 59 FPS	10	-
Wheel encoder	RLS	LM13	Magnetic rotary encoder	100	4096 (resolution)
VRS GPS	SOKKIA	GRX2	VRS-RTK GPS	1	H:10mm, V:15mm
FOG	KVH	DSP-1760	Fiber optics gyro	1000	0.05/h

5.2 provides the parameters used for training the GP. The GP model was constructed using Pyro [9] library for deep probabilistic programming and PyTorch. The training time to execute 100 epochs was 141.66 seconds. This is depicted as demonstrated in Figure 5.5 where training time vs each epoch is illustrated. In addition, the training loss minimization in Figure 5.4 is shown to illustrate the propagation of and settling of the model around epoch 100. In addition, figure 5.5 illustrates the training time utilized to process the entire dataset through the training loop each epoch. The initial spike shown here can be inferred to data generation and caching of the dataset during initial loading.

Table 5.2: Parameters for training GP

Training parameter	value
Optimizer	Adam
Learning rate	0.01
No. of epochs	100
sample rate	100 Hz

Given the posterior predictive distribution mean vector and covariance matrix as obtained in Eqs.(5.15), (5.16) the log marginal likelihood to be maximized conditioned to the hyperparameters $\Theta = (\sigma_k, l)$ can be written as

$$\log p(\Delta \mathbf{X}_i | \Theta) = -\frac{1}{2} \log \det (\mathbf{K}(\bar{\mathbf{U}}_i, \bar{\mathbf{U}}_i) + \sigma_{w_i}^2 \mathbf{I}) - \frac{1}{2} \Delta \mathbf{X}_i^T (\mathbf{K}(\bar{\mathbf{U}}_i, \bar{\mathbf{U}}_i) + \sigma_{w_i}^2 \mathbf{I})^{-1} \Delta \mathbf{X}_i - \frac{n}{2} \log 2\pi, \quad (5.20)$$

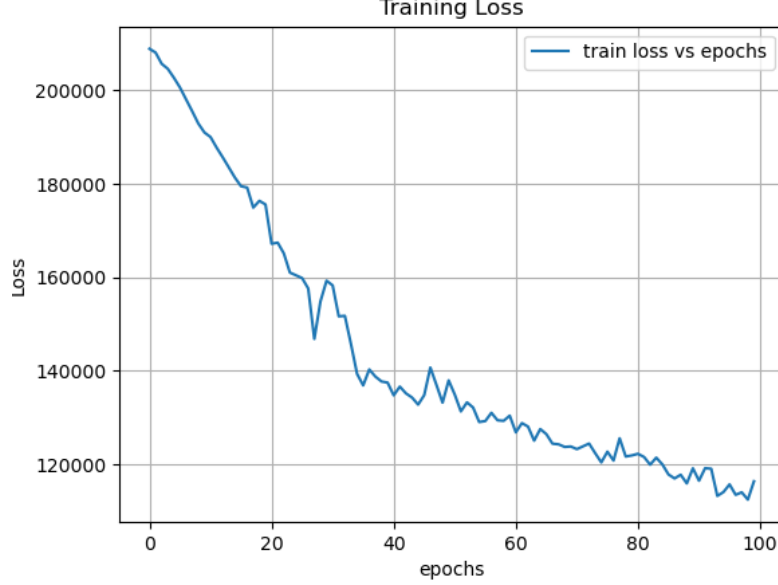


Figure 5.4: Training loss propagation for each epoch illustrating model convergence.

where, $\frac{n}{2}\log 2\pi$ is the normalization constant, $\mathbf{K}(\bar{\mathbf{U}}_i, \bar{\mathbf{U}}_i) \in \mathcal{R}^{n \times n}$ is the kernel matrix where the element (i,j) of $\mathbf{k}_{ij} = \mathbf{k}(u_i, u_j)$, $\mathbf{k}(\cdot, \cdot)$ is the kernel function which consists of the base kernel $\bar{\mathbf{k}}$.

However, given the closed form representation of the Eqs. (5.15), (5.16), (5.20) for very large datasets it is computationally very expensive as the bottleneck here is the storage $O(n^2)$ and inversion $O(n^3)$ of the Kernel matrix as seen on the term $(\mathbf{K}(\bar{\mathbf{U}}_i, \bar{\mathbf{U}}_i) + \sigma_{w_i}^2 \mathbf{I})^{-1}$. Hence, to avoid this cost we approximate the posterior based on m induced points where $m \leq n$.

5.6.2 Results

We ran two **MSCKF**'s, one with and one without slip compensation, to demonstrate the properties of the proposed scheme. To demonstrate the improvements by the proposed scheme, we compare the mean absolute trajectory error (MATE) and mean absolute rotation error (MRTE) between the slip-compensated VIWO and VIWO obtained without slip compensation.

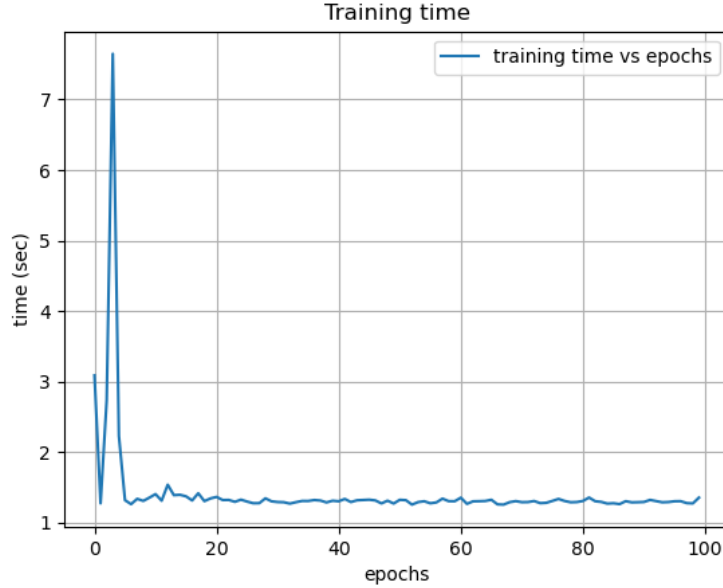


Figure 5.5: Training time at each epoch to complete one cycle of training and optimization.

In Figures 5.6 and 5.7 the robot trajectories depict VIWO with corrected WO and VIWO without slip compensated WO. It is shown that the accuracy of the VIWO scheme with integrated slip compensation is closer to the ground truth of the test sequences. This is quantitatively displayed in Figures 5.8 and 5.9. The red color bar plot shows the MATE/MRTE for VIWO without slip compensation correction while the green color bar plot shows the MATE/MRTE for VIWO with slip compensation correction. As depicted in the error bar plots when slip compensation is integrated into VIWO there is a significant reduction in MATE and MRTE.

We further extend our real-world data based simulation study where we analyse the impact of the DNN architecture of using CNN vs LSTM as illustrated in Figures 5.10, 5.11. Our experimental investigations revealed that LSTM layers can better capture the sequential correlations of the odometry data as shown in the error bar plots as given in Figure 5.12.

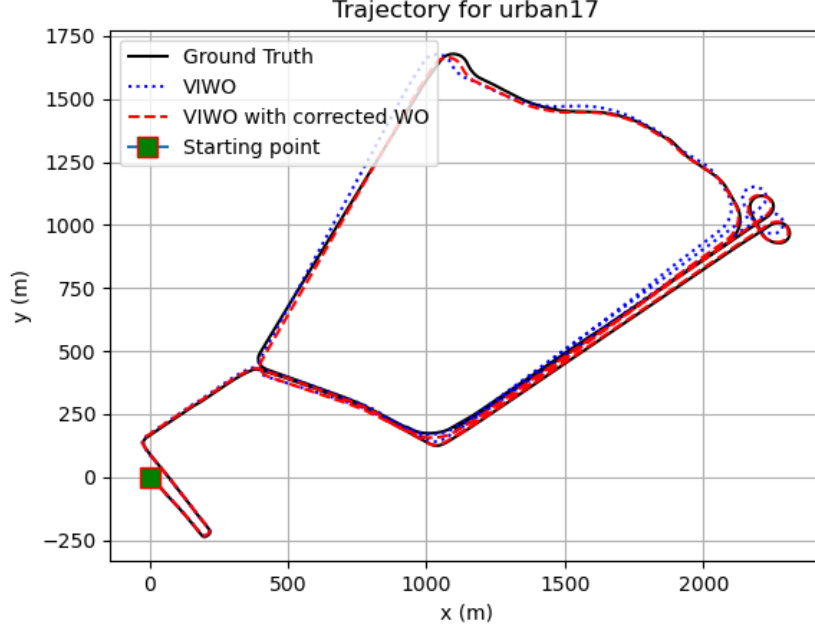


Figure 5.6: Slip compensation comparison for test sequence 1.

5.7 Indoor Experimentation

We validate our proposed methodology by conducting experiments in an indoor setup. We implement and test the scheme using Clearpath’s Jackal UGV (unmanned ground vehicle), equipped with readily available sensors such as wheel encoders and IMU. We also attach an Intel realsense T265 camera to obtain visual data. The Jackal UGV is a skid steer differential drive robot that consists of four wheels where each pair of the wheels on the left side and the right side are actuated by a single motor. Hence, for modeling it can be modeled as a standard two-wheel differential drive system. The sensor specifications and associated measurements can be found in Table 5.3. The data from the Jackal UGV is obtained via ROS (robot operating system) [92]. To validate the experimental results, we compare them with the data obtained from the OptiTrack motion capture system, which serves as the ground truth reference.

Figure 5.13 illustrates the difference in wheel speeds when turning and traversing in an arc and the associated slippage. The aggressive spikes to be seen in the wheel speed measurements are due to the discrete nature of the measurements from the wheel encoder,

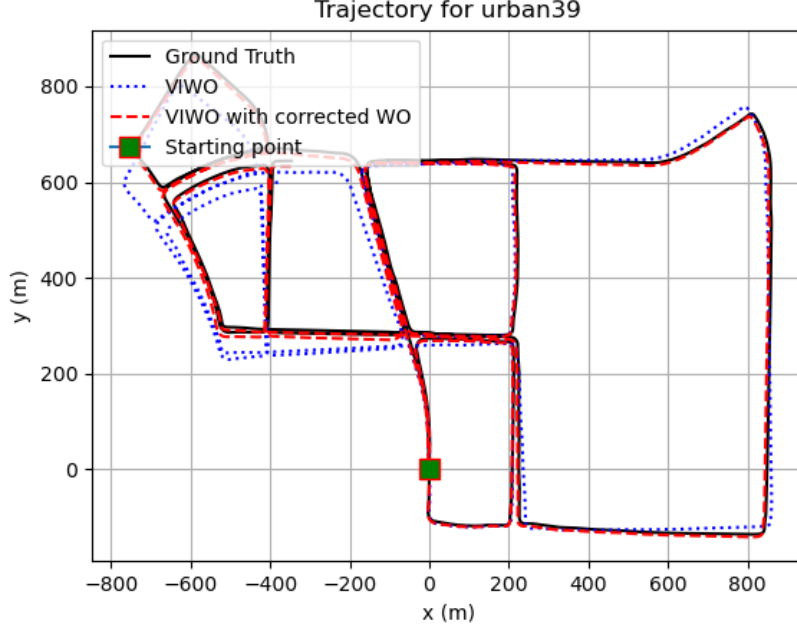


Figure 5.7: Slip compensation comparison for test sequence 2.

further exacerbated by wheel slippage. The measurements are filtered using a moving average filter. In addition, the same Figure illustrates the comparison of the robot trajectory and estimated robot trajectory using the wheel speed measurements showcased. Here we can see the robot trajectory drift due to wheel slippage, where the red color line is the ground truth and the blue color line is the estimate.

Figure 5.14 shows the test setup with the Clearpath UGV Jackal and the Optitrack motion capture system, which provided the ground truth measurements. With the indoor set-up, we tested the proposed algorithm under the same metrics as in section 5.6. Figures 5.15, 5.16 depict the robot trajectories for the cases which include slip compensation plotted against the ground truth trajectory obtained with the optitrack motion capture system. Furthermore, the Figure 5.12 displays the bar plot for RMSE(m) values of the test sequences. From this bar plot, it is evident that the trajectory obtained via augmented slip compensation and dynamic feature point elimination had the lowest RMSE. In summary, our methodology was able to increase the accuracy of localization for the outdoor real-world data as well as the indoor experiments conducted using a UGV.

To further test our methodology, we used gravel and sand on the test floor to determine

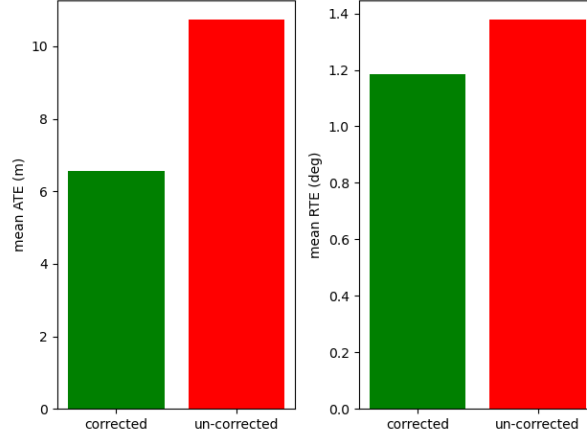


Figure 5.8: Trajectory error for test sequence 1.

Table 5.3: Jackal Sensor specifications.

Sensor	Description	freq.(Hz)
Camera	Intel realsense T265	10fps
IMU		400
Wheel encoders	78,000 pulses/m quadrature	100

the effectiveness of the proposed system. Figure 5.18 shows the robot trajectories when the navigating surface is layered with gravel and sand. Our proposed scheme was able to improve the localization accuracy by aiding in slippage compensation. However, this also shows that in order to apply the developed method in new terrains and surfaces needs retraining of the model to calibrate the model parameters.

5.8 Summary and Conclusion

chapter presented a **VIWO** scheme that incorporates compensation of errors due to noisy measurements from wheel encoders, mainly due to slippage. In this scheme, the error in **WO** between the physical model and ground truth is estimated using a learning based **GPR** model that is trained using a set of training data. Then the learned **WO** error estimate is used to correct the **WO** measurements. The corrected **WO** measurements are used to update an **MSCKF** based **VIWO** state estimation scheme.

As future directions of this work, addition of online calibration of intrinsic and extrinsic

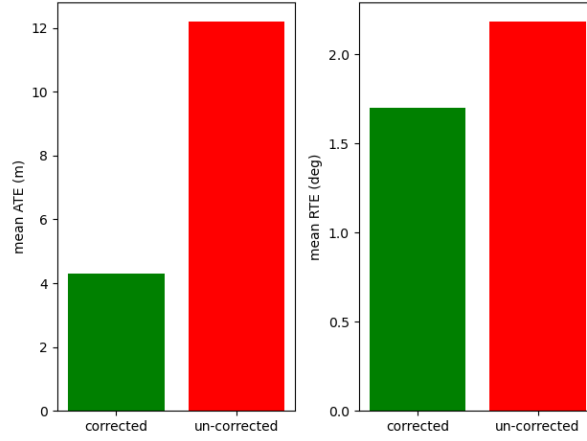


Figure 5.9: Trajectory error for test sequence 2.

parameters of the wheel encoders can be considered. This extension is essential especially when the robot is navigating in harsh environments where parameters such as wheel radius change with time, reducing the localization accuracy. In order to develop more generalized deep GP based methods, the designs are aimed to be tested in other relevant datasets as well as perform real-world experiments. Finally, since the proposed methodology requires supervised learning, further research is needed to explore unsupervised learning approaches to make the design more suitable for real-world applications.

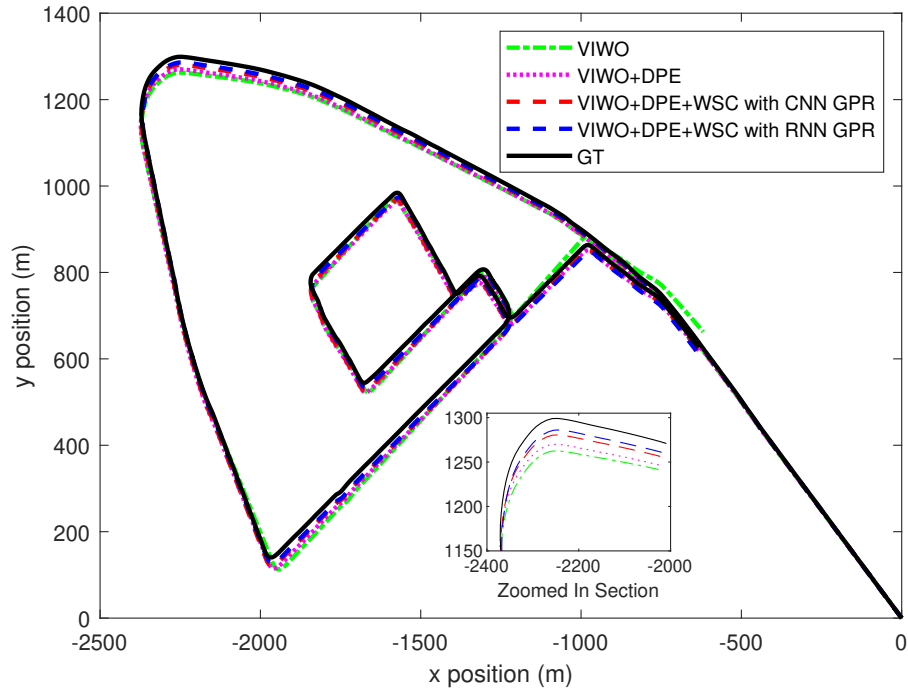


Figure 5.10: Trajectory plot for test sequence 1 with LSTM based Kernel. VIWO : Visual Inertial Wheel Odometry, DPE: Dynamic Point Elimination, WSC: Wheel Slip Compensation.

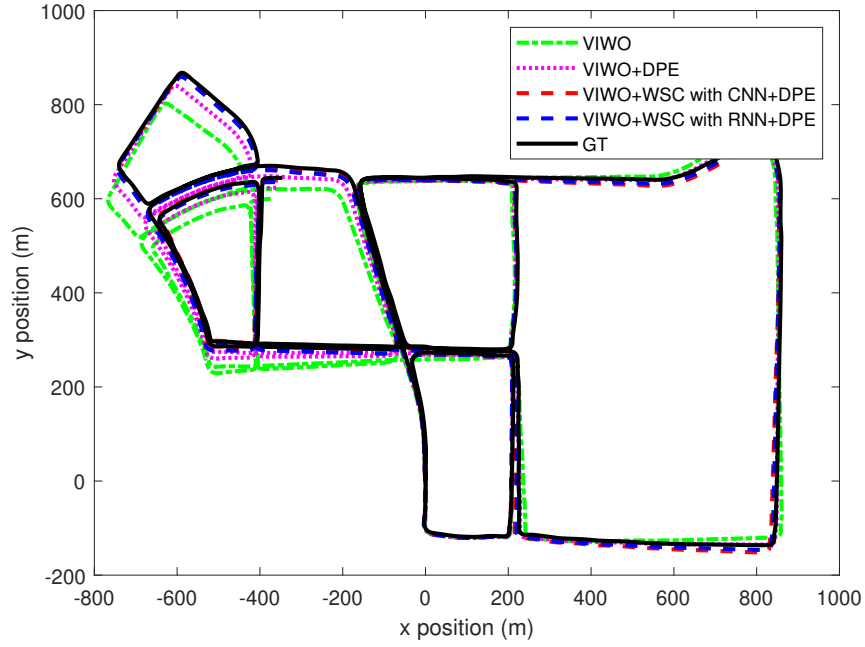


Figure 5.11: Trajectory plot for test sequence 2 with LSTM based Kernel. VIWO: Visual Inertial Wheel Odometry, DPE: Dynamic Point Elimination, WSC: Wheel Slip Compensation.

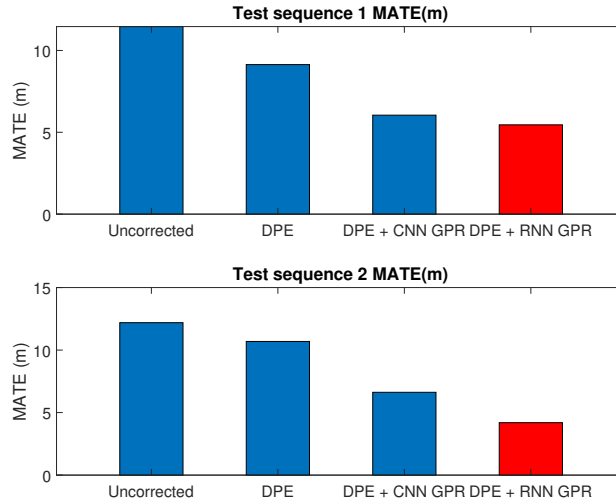


Figure 5.12: Trajectory error for test sequences with LSTM based deep Kernel.

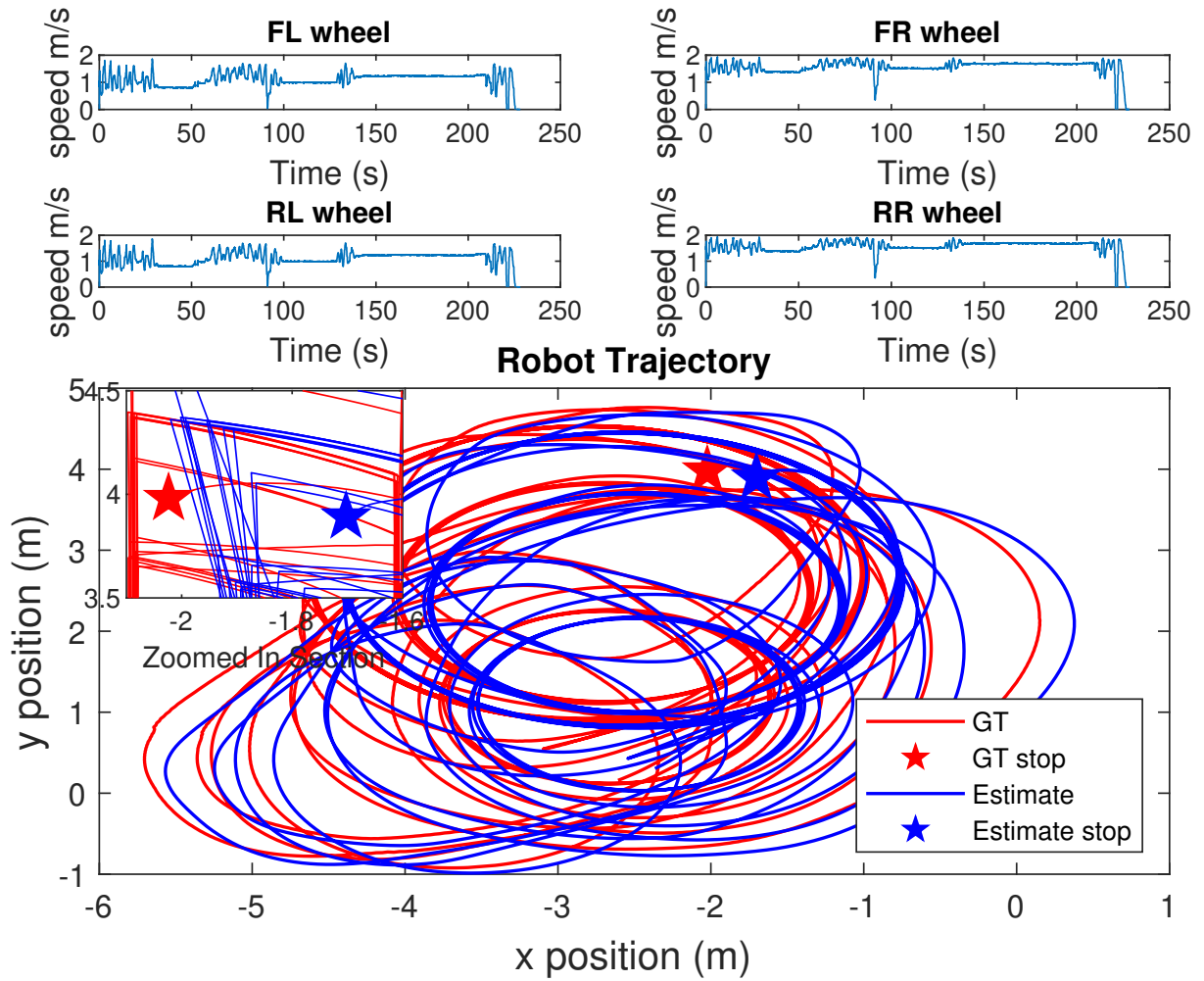


Figure 5.13: Comparison of wheel speeds of the robot along with ground truth captured and trajectory estimated for a sample of training dataset. FL: Front Left, FR: Front Right, RL: Rear Left, RR: Rear Right.

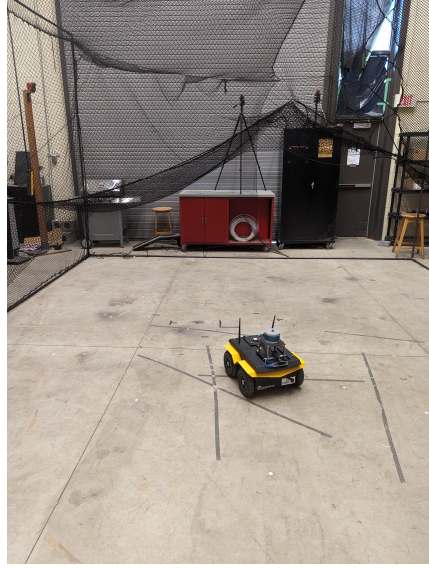


Figure 5.14: Experimental setup, which includes the UGV and Optitrack Motion Capture system.

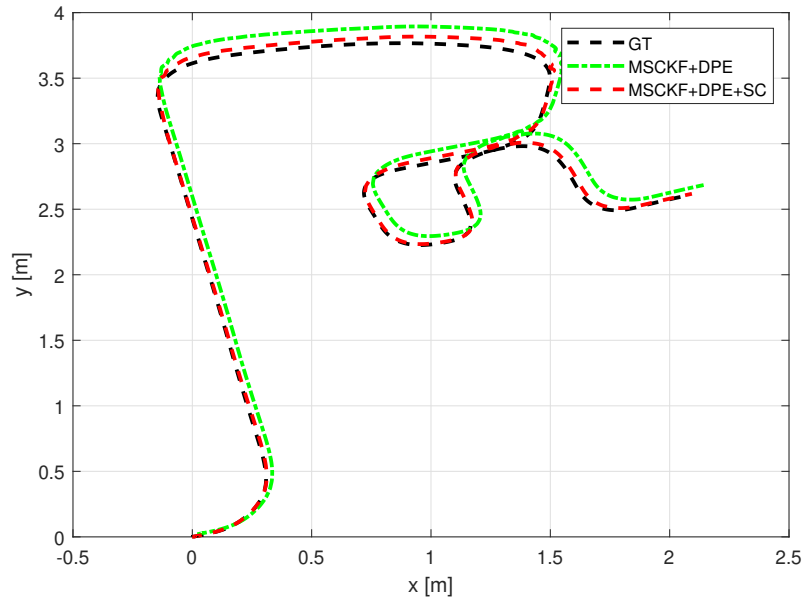


Figure 5.15: Experiment sequence 1 odometry trajectory comparison. DPE: Dynamic Point Elimination, SC: Slip Compensation (LSTM GPR)

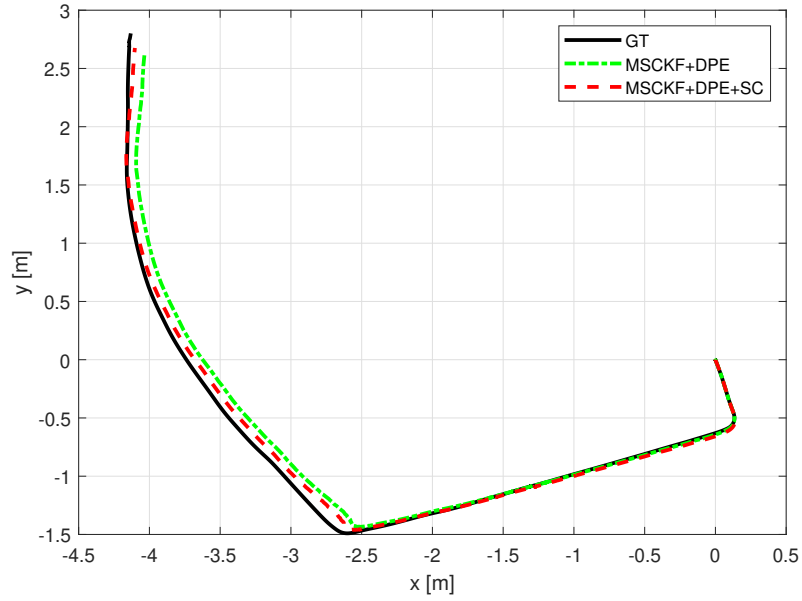


Figure 5.16: Experiment sequence 2 odometry trajectory comparison. DPE: Dynamic Point Elimination, SC: Slip Compensation (LSTM GPR)

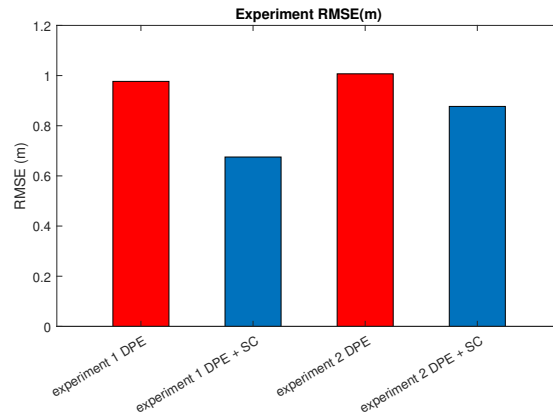


Figure 5.17: RMSE(m) for the experimental trajectories.

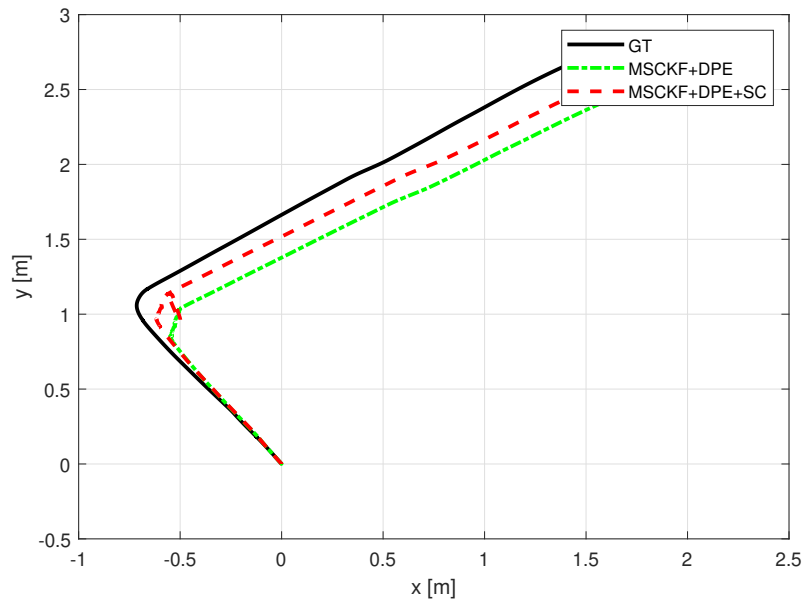


Figure 5.18: Experiment sequence 3 odometry trajectories comparison in gravel/sand based terrain.

Chapter 6

Semantic Segmentation for Navigation

6.1 Introduction

Traditional visual odometry techniques, which often rely on feature matching and optical flow, can be significantly affected by dynamic objects within the scene, leading to inaccuracies in motion estimation. Semantic segmentation, the task of assigning a class label to each pixel in an image, offers a solution by enabling the identification and exclusion of dynamic objects from the data used for odometry calculations. Early developments and foundational Works for the integration of semantic segmentation into visual odometry began with the application of classic machine learning techniques for image segmentation, combined with feature-based odometry algorithms. These initial efforts laid the groundwork for understanding the benefits of semantic information in improving odometry in dynamic scenes. Notably, the advent of deep learning and, specifically, the introduction of architectures like FCN [87](Fully Convolutional Networks) and later, U-Net [121], significantly advanced the field of semantic segmentation, providing the tools necessary for more accurate and efficient dynamic point elimination.

Deep learning enhanced in how semantic information could be leveraged for visual odometry. Convolutional Neural Networks (CNN) became the de facto standard for semantic segmentation tasks, with architectures such as SegNet [6], DeepLab [25], and PSPNet [88] offering substantial improvements in segmentation accuracy. Parallel to these developments, deep learning-based visual odometry systems, such as DeepVO [144], demonstrated the potential of end-to-end learning for motion estimation from raw image sequences. Com-

binning these two advancements, researchers began to explore the explicit use of semantic segmentation for dynamic point elimination in visual odometry systems. Techniques varied from straightforward masking of dynamic objects based on segmentation maps to more sophisticated methods where semantic information directly influenced the odometry estimation process.

A critical methodology that emerged was the use of semantic segmentation to inform feature selection and tracking in visual odometry pipelines. By identifying and ignoring features associated with dynamic objects, such as vehicles and pedestrians, systems could achieve more stable and accurate motion estimates. This approach necessitated advancements in real-time semantic segmentation to keep pace with the requirements of visual odometry systems. Another notable innovation was the development of SLAM systems that integrated semantic understanding, such as SemanticFusion and DynaSLAM [8]. These systems demonstrated the dual benefits of semantic segmentation: improving the map’s accuracy by excluding dynamic objects and enhancing the system’s understanding of the environment by classifying structural elements.

Considering the aforementioned challenges and advancements, this chapter provides an additional add-on filter based on semantic segmentation as an alternative to the work done in chapter 4.

6.2 Overall Methodology

The overall structure of the proposed scheme is discussed in this section. The work described in this section builds on top of Algorithm 2 discussed in Chapter 4. We use semantic segmentation as an additional filter to assist in dynamic feature point elimination.

Figure 6.1 shows the block diagram with key components of the proposed system. These key components can be summarized as in the following. The MSCKF takes measurements from camera and IMU. The camera measurement function given in eq.(4.5) obtains the feature measurements as given in eq.(4.6). The inertial measurements are preintegrated as given in eqs.(4.26),(4.27) to match the frequency of visual measurements to obtain translation and rotation of the inertial frame ${}^I(\cdot)$. The acquired position and rotation is then used to construct an estimate of the essential matrix. Since the internal parameter matrix is known, the estimated essential matrix can be transformed in an estimate of the fundamental matrix. This estimated fundamental matrix is then used to check the feature confidence of features extracted from segmented potential dynamic objects to classify feature points as dynamic or static based on the condition given in eq.(4.36). Once the pre-processed feature

points are obtained after dynamic feature elimination they are marginalized to arrive at the final measurement function eq. (4.8) which is used directly in [MSCKF](#).

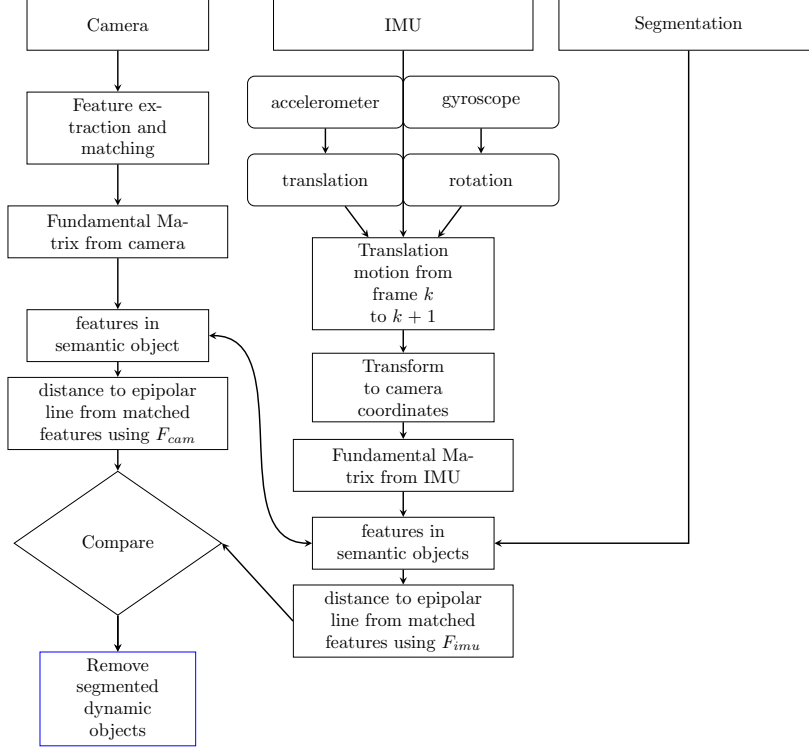


Figure 6.1: Overall structure of dynamic object removal

For the purpose of semantic segmentation, we utilize Mask R-CNN as the main segmentation algorithm. Mask R-CNN [49] is an advanced and highly efficient model for instance segmentation, enhancing the foundational Faster R-CNN [120] framework. The working mechanism of Mask R-CNN is outlined in figure 6.2. It incorporates five key components essential for accurate instance segmentation: 1- the backbone network, 2- the region proposal network (RPN), 3- the Region of Interest Align (RoIAlign) layer, 4- the detection head, and 5- the mask head. These components are visualized in the structure of the Mask R-CNN model, as shown in figure 6.2.

The backbone network, typically a deep CNN like ResNet, forms the core of Mask R-CNN, processing input images to extract detailed feature maps that contain rich semantic and spatial information. The RPN’s main function is to identify potential bounding boxes around objects, which are refined and classified in later stages. It does so by analyzing the

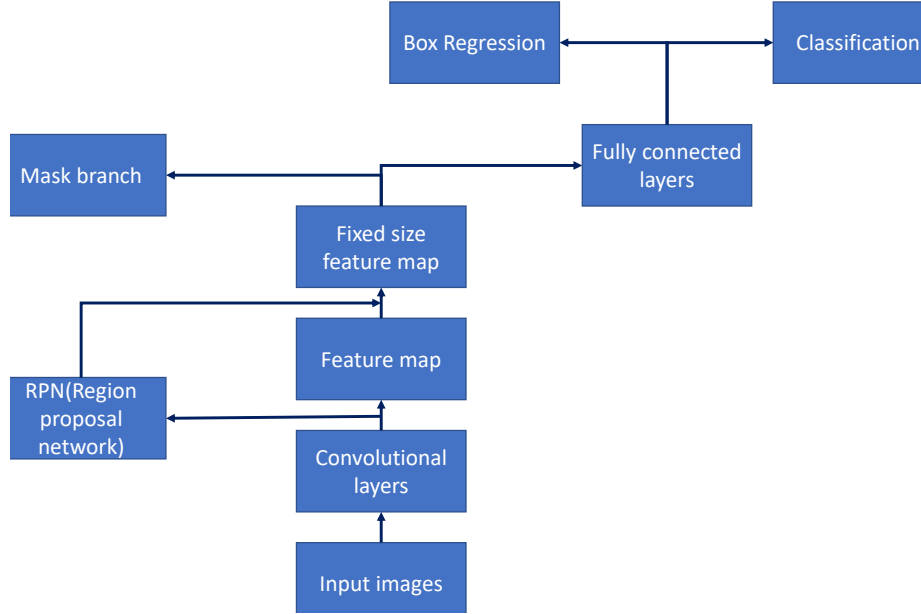


Figure 6.2: Outline overview of Mask-RCNN network [50]. (a) In the first stage there are two networks. a backbone network (Resnet in this article) and a RPN (region proposal network). Region proposals are regions in the feature map which consists an object. (b) The second stage consists of the prediction of object class and bounding boxes

features from the backbone network, utilizing anchor boxes of various sizes and ratios to densely scan the image. For each anchor box, the RPN calculates objectness scores and bounding box adjustments, using classification and regression losses to improve proposal accuracy. Upon generating region proposals, the features and proposals go through the region of interest align layer, which precisely extracts features for each proposal by employing bi-linear interpolation. These features are then processed by the detection and mask heads to predict object classes and generate binary masks, highlighting objects within the proposals.

Throughout its training phase, Mask R-CNN leverages ground truth annotations to refine its predictions across class identification, bounding box adjustments, and mask generation. In its inference mode, the model uses the backbone network to derive feature maps from an input image, then the RPN to create and refine region proposals. These proposals are further processed to produce accurate class identifications and instance masks, enabling precise segmentation of objects within the image.

For training the mask-RCNN architecture we use the Cityscapes dataset [27] which is



Figure 6.3: City scene dataset example segmentation.

developed for semantic urban scene understanding. Figure 6.3 illustrates a sample segmentation based on training dataset, while Figure 6.4 illustrates sample segmentation results for the test dataset consisting of dynamic objects such as moving vehicles and pedestrians.

6.3 Dynamic feature point elimination

To differentiate if a segmented object is dynamic or static, we use an IMU model to check if a dynamic object identified via semantic segmentation is static or moving in the environment, as given in Algorithm 3. This algorithm differentiates from Algorithm 2 perspective, where the input features to the algorithm is further filtered using semantic segmented masking.

The computed translation and rotation from the process model can also be used for calculating the corresponding fundamental matrix $\mathbf{F}'$ using the camera-IMU extrinsic parameters, which then be used in the dynamic point elimination method [119]. A fundamental matrix $\mathbf{F}$ can also be calculated using the IMU data. With both fundamental matrices, a distance measure between the epipolar line and the corresponding features using the both matrices can be computed and denoted as $d_{k,i}^1$ and $d_{k,i}^2$, respectively as given as

$$[x, y, z]^\top = \mathbf{F}_{f(i)}^c \mathbf{p}_k, [x', y', z']^\top = \mathbf{F}'_{f(i)}^c \mathbf{p}_{k+1} \quad (6.1)$$

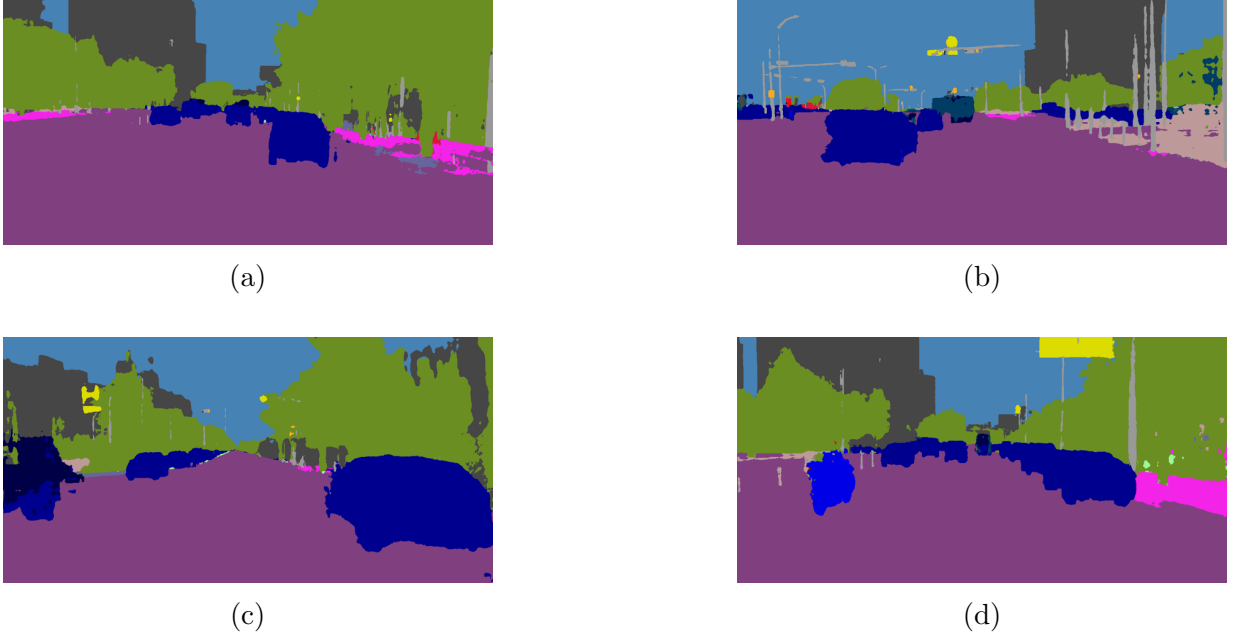


Figure 6.4: Sample segmentation results on dynamic scenes from the test dataset

$$d_{k,i}^1 = \frac{|{}_{f(i)}^c \mathbf{p}_{k+1}^\top \mathbf{F} {}_{f(i)}^c \mathbf{p}_k|}{\sqrt{||x||^2 + ||y||^2}} \quad (6.2)$$

$$d_{k,i}^2 = \frac{|{}_{f(i)}^c \mathbf{p}_k^\top \mathbf{F}' {}_{f(i)}^c \mathbf{p}_{k+1}|}{\sqrt{||x'||^2 + ||y'||^2}} \quad (6.3)$$

As the next step a threshold ρ is used to determine whether each matched feature point (after discarding the keypoints from segmented dynamic objects) is static or dynamic as given in Eq.(4.36). $\phi_{k,i}$ in Eq.(4.36) indicates the consistency between the i^{th} feature point of the k^{th} frame and the fundamental matrix. $\phi_{k,i}$ is set to 0 if the feature point is consistent, and 1 if otherwise. Additionally, there can be a case when IMU data is invalid for the elimination method. To combat this issue, we use an additional threshold μ to keep track of match count, as given in Eq. (4.37). Hence, the dynamic point elimination will only use the IMU data if and only if the data is valid (i.e $\psi_k = 0$) and trustworthy. Alternatively, if the confidence in the IMU-based dynamic point elimination is insufficient, the system switches to a [RANSAC](#)-based outlier rejection scheme with threshold ζ as given in Eq.(4.40).

Figure 6.1 explains removal of dynamic feature points using semantic segmentation.

Algorithm 3: Algorithm for dynamic point elimination

Input : matched feature points of adjacent frames
 $\Pi_k = \{\mathbf{p}_{k,1}, \dots, \mathbf{p}_{k,n}\}, \Pi_{k+1} = \{\mathbf{p}_{k+1,1}, \dots, \mathbf{p}_{k+1,n}\}$

Input : masked dynamic objects

Input : matched feature points of adjacent frames based on the masked dynamic objects $\Pi_k = \{\mathbf{p}_{k,1}, \dots, \mathbf{p}_{k,m}\}, \Pi_{k+1} = \{\mathbf{p}_{k+1,1}, \dots, \mathbf{p}_{k+1,m}\}$ where $m \leq n$

Output: refined feature points with dynamic feature points removed

while $t \leq t_{image}$ **do**

for each point $\mathbf{p}_{k,i} \in \Pi_k$ **and** $\mathbf{p}_{k+1,i} \in \Pi_{k+1}$ **do**

$[x, y, z]^\top = \mathbf{F}\mathbf{p}_{k,i}$;

$d_{k,i}^1 = \frac{|\mathbf{p}_{k+1,i}^\top \mathbf{F}\mathbf{p}_{k,i}|}{\sqrt{\|x\|^2 + \|y\|^2}}$;

$[x', y', z']^\top = \mathbf{F}'\mathbf{p}_{k+1,i}$

$d_{k,i}^2 = \frac{|\mathbf{p}_{k,i}^\top \mathbf{F}'\mathbf{p}_{k+1,i}|}{\sqrt{\|x'\|^2 + \|y'\|^2}}$;

if $\psi_k = 1$ **and** $|d_{k,i}^1 - d_{k,i}^2| \geq \rho$ **then** remove dynamic points $\mathbf{p}_{k,i}, \mathbf{p}_{k+1,i}$ from Π_k, Π_{k+1} ;

else if $\psi_k = 0$ **then**

 then remove outliers according to RANSAC with threshold ζ as given in Eq.(4.40)

end

end

end

We check a sample of extracted features from an object are dynamic or static and use that information to conclude whether to discard or keep the features associated with the segmented object. Figure 6.5 illustrates an example of eliminating dynamic feature points. Two moving objects which are cars can be seen in subsequent frames. Once the segmentation algorithm detects these moving objects and confirms its dynamic nature with IMU based as shown in this figure, the extracted feature points are eliminated from the matching process.

Algorithm 3 describes the intricate logic flow of the developed dynamic point elimination scheme. Algorithm 3 differs from Algorithm 2 as we only check a subset of feature points as the Input from detected dynamic objects to arrive at conclusions with regard to the whole set of features extracted from that dynamic object. We estimate the fundamental matrix

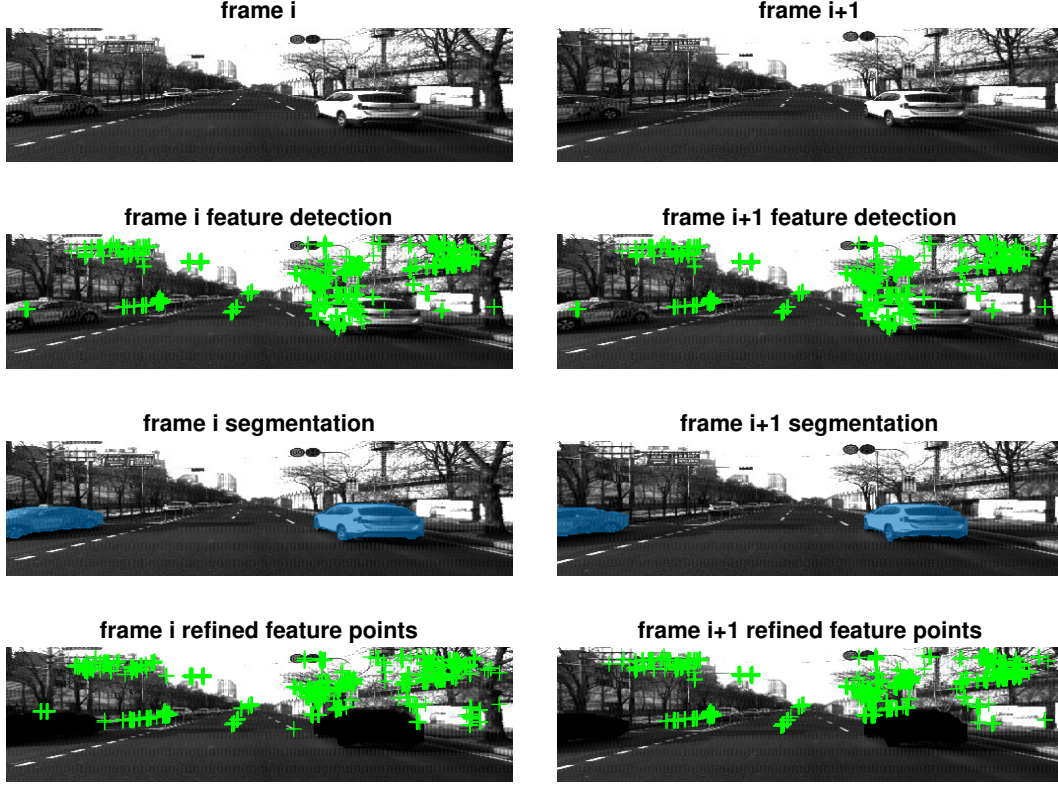


Figure 6.5: Dynamic point removal, real world data example using semantic segmentation and geometric consistency.

$\mathbf{F}'$ using the IMU motion model. Similarly, the Fundamental matrix relating two images $\mathbf{F}$ is obtained. Two distance measures $d_{k,i}^1, d_{k,i}^2$ as given in algorithm 2 and Eqs.(6.2),(6.3) are used to classify the feature points in dynamic objects as static or dynamic. Visual measurements are obtained as in Eq.(4.5) discarding the dynamic object features and are used to correct the inertial propagated dynamic model via MSCKF.

6.4 Real-World Data Based Simulations

In this section, we provide experimental results obtained through simulations over publicly available datasets. We demonstrate the approach using the KAIST [58] data-set which is collected in an urban area. Table 5.1 provides the KAIST dataset sensor specifications.

The real world data based simulations were carried out on a Intel Core i7 laptop equipped with a NVIDIA GeForce GTX 1660 Ti GPU. Figures 6.6 and 6.9 illustrates the benefits of combining semantic segmentation with geometric feature confidence estimation to effectively eliminate dynamic feature points on visual front end applications.

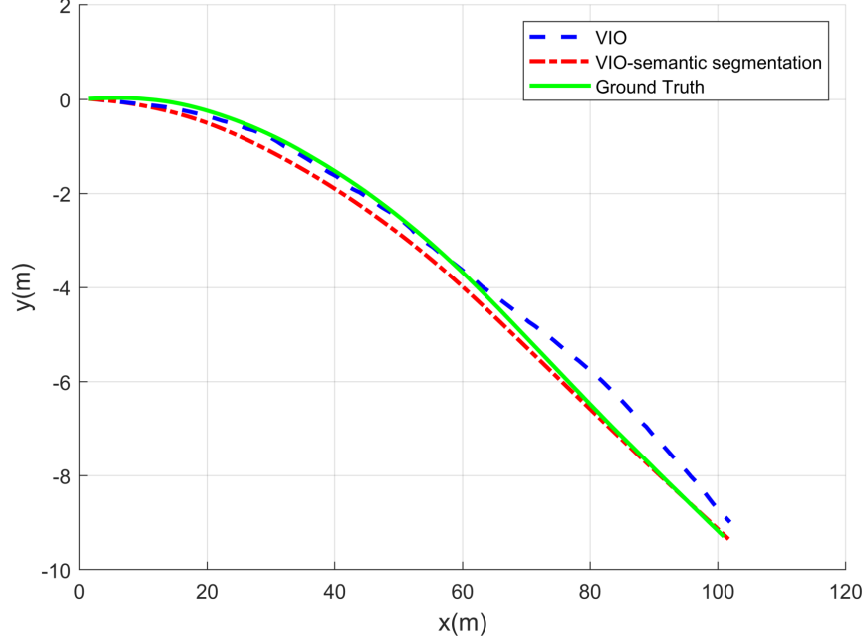


Figure 6.6: Trajectory 1 localization estimation with and without Semantic Segmentation.

The real world data simulations studies reveal that it is possible to improve the localization accuracy by pre-identifying the dynamic objects and culling only the associated dynamic points of dynamic objects. As illustrated in table 6.1 when the dynamic point elimination is combined with semantic segmentation iterates to better culling of dynamic points and outliers. We provide sample runs on two trajectory profiles, which illustrates the benefits of combining semantic segmentation with our developed geometric feature classification technique using the IMU motion model. The trajectories in the first case when semantic segmentation is considered closely follows the reference trajectory indicating an average RMSE of 0.4807 m, a reduction from the average RMSE of 1.9898 m. This is similar in nature to the second case, where an average RMSE reduction from 0.6728 m to 0.3352 m was achieved. Figures 6.7,6.8 illustrates the propagation of translation error (m) and rotation error (rad) for Trajectory 1. Figures 6.10,6.11 illustrates the propagation of translation error (m) and rotation error (rad) for Trajectory 2, respectively.

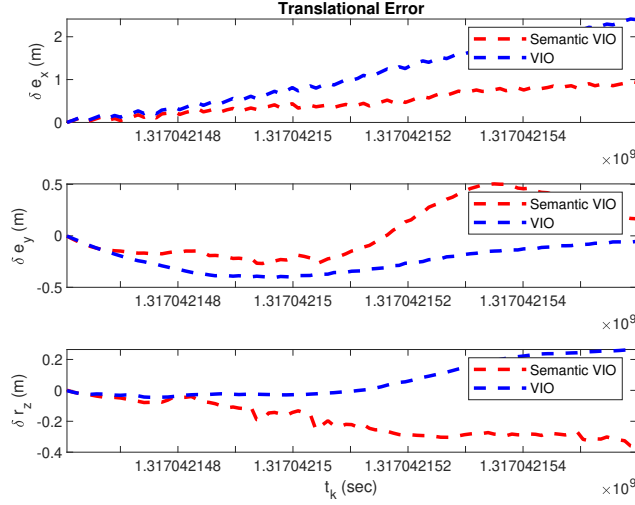


Figure 6.7: Trajectory 1 translation error plot in comparison to with and without Semantic Segmentation.

ARMSE comparison			
with Semantic segmentation		without semantic segmentation	
Translation (m)	rotation (rad)	Translation (m)	Rotation (rad)
0.4807	0.0112	1.9898	0.018
0.3352	0.0025	0.6728	0.0099

Table 6.1: Translation error comparison for with and without Semantic Segmentation implementation depicted in ARMSE (average root mean squared error).

From the translation error plots, it can be seen that it is consistently fluctuating around the desired zero level. Significant drifts can be seen in relation to the translation errors when a dynamic point elimination scheme is not considered. The second trajectory rotation error plot closely follows the pattern of its translation error plot counterpart. However, discrepancies can be observed for the second trajectory rotation error plot, exacerbating the orientation estimation accuracy when using Semantic Segmentation. This can be inferred to the existing wheel slippage and drift in orientation estimation.

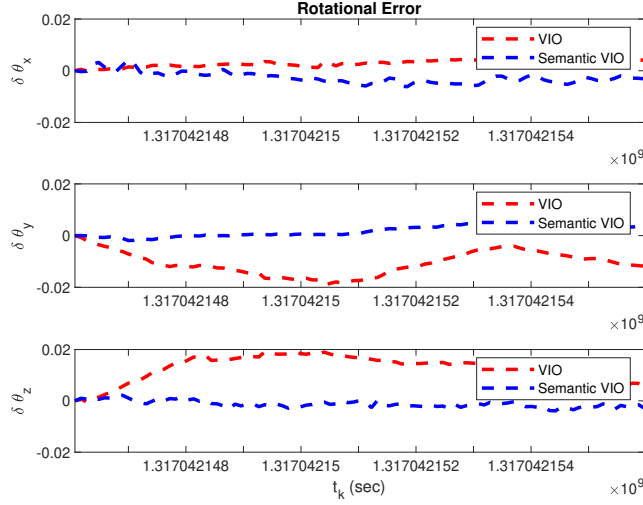


Figure 6.8: Trajectory 1 rotation error plot in comparison to with and without Semantic Segmentation.

6.5 Summary and Conclusion

Semantic segmentation presents a powerful alternative to [RANSAC](#) for dynamic point elimination in visual odometry systems. By providing detailed scene understanding, it enables more accurate and robust motion estimation in dynamic environments. However, realizing its full potential requires addressing current limitations, particularly regarding real-time performance and model generalization. As research in this area progresses, the integration of semantic segmentation into [VO](#) and broader computer vision tasks promises to unlock new capabilities for autonomous systems, enhancing their ability to navigate and understand complex, ever-changing environments.

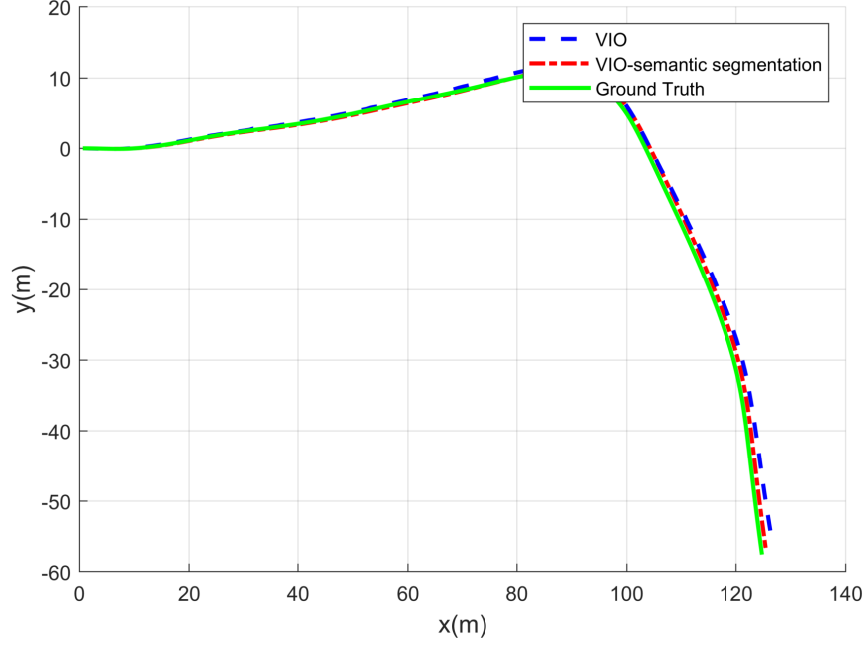


Figure 6.9: Trajectory 2 localization estimation with and without Semantic Segmentation.

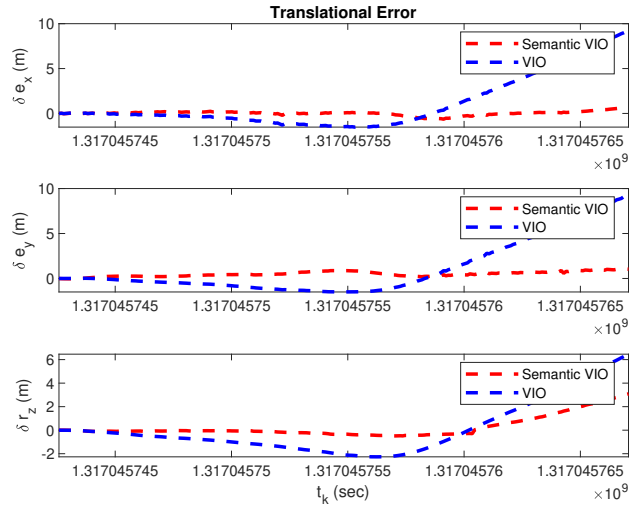


Figure 6.10: Trajectory 2 translation error plot in comparison to with and without Semantic Segmentation.

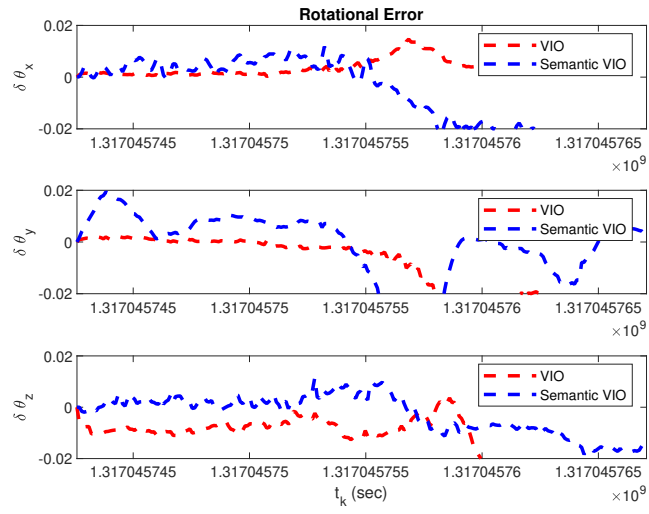


Figure 6.11: Trajectory 2 rotation error plot in comparison to with and without Semantic Segmentation.

Chapter 7

Conclusion and Future Work

7.1 Conclusions

The main objective of this thesis was to design accurate localization algorithms for mobile robotic agents. In the event of unavailability and/or degradation of GNSS systems, a design for estimating robot pose for localization was proposed using commonly available sensory information for a typical mobile robot. These information included visual, inertial and wheel encoder measurements. An MSCKF based filtering scheme was designed to fuse the sensor information. Two main challenges were addressed to improve the localization accuracy: (a) how to eliminate dynamic points in feature matching for VO front end to improve overall localization accuracy, and (b) how to compensate for WO to improve overall localization accuracy.

To eliminate dynamic feature points associated with dynamic objects, a geometric feature point confidence estimator was designed using IMU data to check the consistency of feature points. It was shown that the developed dynamic feature point confidence estimator can effectively eliminate dynamic feature points. In addition, the design also takes into consideration the validity of the IMU data that was used for estimating the confidence and then switch the outlier removal mechanism back to a RANSAC based mechanism. The designed strategy can be used as a layer to RANSAC and improve its performance. However,

A slip compensator design was also carried out using GPR based estimation of the odometry residuals. Specifically, a GPR scheme was designed using deep kernels: (a) using 1D CNN layers, and (b) using LSTM layers. Based on real-world data simulations, it was

observed that [LSTM](#) was slightly better than [CNN](#) for learning embedded temporal correlations associated with [WO](#) residuals. Experimental evaluations showed that the data driven methodology works well in encapsulating the [WO](#) error residuals and thereby compensating it. It is to note that this methodology is constrained by data availability. Hence, for true adaptable autonomous navigation, one must find ways to implement unsupervised learning based methods/mechanisms.

7.2 Future Work and Directions

This section summarizes key challenges still pending which could converge the research direction.

Despite its advantages, the application of [GPR](#) in slip compensation faces several challenges. One major issue is the computational complexity of [GPR](#), especially as the dataset size increases, which can limit its use in real-time applications. The most computationally intensive part of GPR is the inversion of the covariance matrix, which is a $N \times N$ matrix, where N is the number of training points. The complexity of matrix inversion is typically $O(N^3)$ making the process very expensive for large datasets. Additionally, the performance of [GPR](#)-based models heavily depends on the choice of kernel functions and hyperparameters, requiring extensive tuning and validation.

To address these challenges, recent must be focused on developing more computationally efficient GPR algorithms, such as sparse [GPR](#) and online learning approaches, that are suitable for real-time applications. Furthermore, the integration of [GPR](#) with deep learning techniques, such as deep kernel learning, offers promising avenues for creating more adaptive and accurate slip compensation models. In addition, Incremental or online learning techniques can also be employed, where the model is updated with new data points as they become available, rather than retraining the model from scratch. This approach can be beneficial in wheeled robots navigating in dynamic terrains. In addition, dimension reduction techniques can be used to reduce the dimensionality of the input space, thus reducing the complexity of the model.

Comparing semantic segmentation directly with [RANSAC](#) involves considering accuracy, computational efficiency, and applicability in diverse environments. While semantic segmentation offers superior scene understanding and the ability to inherently distinguish between static and dynamic elements, it requires substantial computational resources, especially for real-time applications. [RANSAC](#), on the other hand, is less computationally demanding but lacks the contextual awareness provided by semantic segmentation. Hybrid

approaches that leverage the strengths of both offer a promising avenue for research and application.

Despite its potential, the use of semantic segmentation for dynamic point elimination faces challenges. The accuracy of semantic segmentation models, particularly in complex or unfamiliar environments, directly impacts the reliability of dynamic point elimination. Real-time performance remains a critical issue, requiring efficient model architectures and hardware acceleration. Future research directions may include the development of lightweight semantic segmentation models optimized for real-time applications, unsupervised and semi-supervised learning approaches to reduce reliance on large labeled datasets, and the exploration of novel neural network architectures that can more effectively integrate geometric and semantic information.

References

- [1] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 10 2023.
- [2] Alexandre Alapetite, Zhongyu Wang, John Paulin Hansen, Marcin Zajczkowski, and Mikołaj Patalan. Comparison of three off-the-shelf visual odometry systems. *Robotics*, 9(3):56, 2020.
- [3] Mary B Alatis and Gerhard P Hancke. Pose estimation of a mobile robot based on fusion of imu data and vision data using an extended Kalman filter. *Sensors*, 17(10):2164, 2017.
- [4] Frank C Anderson and Marcus G Pandy. Dynamic optimization of human walking. *J. Biomech. Eng.*, 123(5):381–390, 2001.
- [5] Ehsan Asadi and Carlo L Bottasso. Tightly-coupled vision-aided inertial navigation via trifocal constraints. In *International Conference on Robotics and Biomimetics (ROBIO)*, pages 85–90. IEEE, 2012.
- [6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.
- [7] Jan Bayer and Jan Faigl. On autonomous spatial exploration with small hexapod walking robot using tracking camera intel realsense t265. In *European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2019.
- [8] Berta Bescos, José M. Fácil, Javier Civera, and José Neira. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, 2018.

- [9] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.
- [10] Karnika Biswas and Indrani Kar. Validating observer based on-line slip estimation for improved navigation by a mobile robot. *International Journal of Intelligent Robotics and Applications*, pages 1–12, 2022.
- [11] Nikolas Brasch, Aljaz Bozic, Joe Lallemand, and Federico Tombari. Semantic monocular SLAM for highly dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 393–400, 2018.
- [12] Martin Brossard and Silvere Bonnabel. Learning wheel odometry and IMU errors for localization. In *International Conference on Robotics and Automation (ICRA)*, pages 291–297. IEEE, 2019.
- [13] 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*, 2016.
- [14] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.
- [15] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European Conference on Computer Vision*, pages 778–792. Springer, 2010.
- [16] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 2021.
- [17] Carlos Campos, José MM Montiel, and Juan D Tardós. Inertial-only optimization for visual-inertial initialization. In *International Conference on Robotics and Automation (ICRA)*, pages 51–57. IEEE, 2020.
- [18] Carlos Campos, José M.M. Montiel, and Juan D. Tardós. Inertial-only optimization for visual-inertial initialization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 51–57, 2020.

- [19] Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of Michigan North Campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35(9):1023–1035, 2016.
- [20] Moises J. Castro-Toscano, Julio C. Rodríguez-Quinonez, Daniel Hernández-Balbuena, Lars Lindner, Oleg Sergiyenko, Moises Rivas-Lopez, and Wendy Flores-Fuentes. A methodological use of inertial navigation systems for strapdown navigation task. In *Proc. IEEE 26th International Symposium on Industrial Electronics*, pages 1589–1595, 2017.
- [21] Le Chang, Xiaoji Niu, and Tianyi Liu. GNSS/IMU/ODO/LiDAR-SLAM integrated navigation system using IMU/ODO pre-integration. *Sensors*, 20(17), 2020.
- [22] Averil B. Chatfield. *Fundamentals of High Accuracy Inertial Navigation*. American Institute of Aeronautics and Astronautics, 1997.
- [23] Averil Burton Chatfield. *Fundamentals of high accuracy inertial navigation*, volume 174. American Institute of Aeronautics and Astronautics, 1997.
- [24] Chang Chen, Hua Zhu, Menggang Li, and Shaoze You. A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives. *Robotics*, 7(3):45, 2018.
- [25] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.
- [26] Shuo Cheng, Liang Li, and Jie Chen. Fusion algorithm design based on adaptive sckf and integral correction for side-slip angle observation. *IEEE Transactions on Industrial Electronics*, 65(7):5754–5763, 2018.
- [27] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [28] Peter Corke, Jorge Lobo, and Jorge Dias. An introduction to inertial and visual sensing. *The International Journal of Robotics Research*, 26(6):519–535, 2007.

- [29] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.
- [30] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.
- [31] Frank Dellaert and Michael Kaess. *Factor Graphs for Robot Perception*. Foundations and Trends in Robotics, Vol. 6, 2017.
- [32] Ziming Ding, Tiankai Yang, Kunyi Zhang, Chao Xu, and Fei Gao. VID-Fusion: Robust visual-inertial-dynamics odometry for accurate external force estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 14469–14475, 2021.
- [33] Tue-Cuong Dong-Si and Anastasios I Mourikis. Estimator initialization in vision-aided inertial navigation with unknown camera-imu calibration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1064–1071, 2012.
- [34] Kevin Ekenhoff, Patrick Geneva, Jesse Bloecker, and Guoquan Huang. Multi-camera visual-inertial navigation with online intrinsic and extrinsic calibration. In *International Conference on Robotics and Automation (ICRA)*, pages 3158–3164, 2019.
- [35] Kevin Ekenhoff, Patrick Geneva, and Guoquan Huang. Closed-form preintegration methods for graph-based visual-inertial navigation. *The International Journal of Robotics Research*, 38(5):563–586, 2019.
- [36] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2017.
- [37] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.
- [38] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [39] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016.

- [40] Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90, 2012.
- [41] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.
- [42] Mingjing Gao, Min Yu, Hang Guo, and Yuan Xu. Mobile robot indoor positioning based on a combination of visual and inertial sensors. *Sensors*, 19(8):1773, 2019.
- [43] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [44] Patrick Geneva, Kevin Eickenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. Opencvins: A research platform for visual-inertial estimation. In *Proc. IEEE International Conference on Robotics and Automation*, pages 4666–4672, 2020.
- [45] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.
- [46] Giorgio Grisetti, Rainer Kümmerle, Hauke Strasdat, and Kurt Konolige. g2o: A general framework for (hyper) graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, pages 9–13, 2011.
- [47] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. <http://eigen.tuxfamily.org>, 2010.
- [48] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [49] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.
- [50] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):386–397, 2020.
- [51] Christoph Hertzberg, René Wagner, Udo Frese, and Lutz Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 14(1):57–77, 2013.

- [52] Joel A Hesch, Dimitrios G Kottas, Sean L Bowman, and Stergios I Roumeliotis. Consistency analysis and improvement of vision-aided inertial navigation. *Transactions on Robotics*, 30(1):158–176, 2013.
- [53] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3946–3952, 2008.
- [54] Jerry Hsiung, Ming Hsiao, Eric Westman, Rafael Valencia, and Michael Kaess. Information sparsification in visual-inertial odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1146–1153. IEEE, 2018.
- [55] Jwu-Sheng Hu and Ming-Yuan Chen. A sliding-window visual-imu odometer based on tri-focal tensor geometry. In *International conference on robotics and automation (ICRA)*, pages 3963–3968. IEEE, 2014.
- [56] Vadim Indelman, Pini Gurfil, Ehud Rivlin, and Hector Rotstein. Real-time vision-aided localization and navigation based on three-view geometry. *Transactions on Aerospace and Electronic Systems*, 48(3):2239–2259, 2012.
- [57] Vladimir G Ivancevic and Tijana T Ivancevic. Lecture notes in lie groups. *arXiv preprint arXiv:1104.1106*, 2011.
- [58] Jinyong Jeong, Younggun Cho, Young-Sik Shin, Hyunchul Roh, and Ayoung Kim. Complex urban dataset with multi-level sensors from highly diverse urban environments. *International Journal of Robotics Research*, 38(6):642–657, 2019.
- [59] Yue Jiang, M Kang, Ming Fan, Sung Chae, and Sung Ko. A novel relative camera motion estimation algorithm with applications to visual odometry. In *Proc. IEEE International Symposium on Multimedia*, pages 215–216, 2018.
- [60] Yue Jiang, Mun-Cheon Kang, Ming Fan, Sung-Ho Chae, and Sung-Jea Ko. A novel relative camera motion estimation algorithm with applications to visual odometry. In *International Symposium on Multimedia (ISM)*, pages 215–216. IEEE, 2018.
- [61] Eunah Jung, Nan Yang, and Daniel Cremers. Multi-frame gan: Image enhancement for stereo visual odometry in low light. In *Conference on Robot Learning*, pages 651–660, 2020.

- [62] Jae Hyung Jung, Jaehyuck Cha, Jae Young Chung, Tae Ihn Kim, Myung Hwan Seo, Sang Yeon Park, Jong Yun Yeo, and Chan Gook Park. Monocular visual-inertial-wheel odometry using low-grade imu in urban areas. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):925–938, 2022.
- [63] Sukwoo Jung, Youngmok Cho, Doojun Kim, and Minho Chang. Moving object detection from moving camera image sequences using an inertial measurement unit sensor. *Applied Sciences*, 10(1):268, 2020.
- [64] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.
- [65] Rong Kang, Lu Xiong, Mingyu Xu, Junqiao Zhao, and Peizhi Zhang. VINS-Vehicle: A tightly-coupled vehicle dynamics extension to visual-inertial state estimator. In *Proc. IEEE Intelligent Transportation Systems Conference*, pages 3593–3600, 2019.
- [66] Marco Karrer, Patrik Schmuck, and Margarita Chli. CVI-SLAM—collaborative visual-inertial SLAM. *IEEE Robotics and Automation Letters*, 3(4):2762–2769, 2018.
- [67] Cagri Kilic, Nicholas Ohi, Yu Gu, and Jason N. Gross. Slip-based autonomous ZUPT through gaussian process to improve planetary rover localization. *IEEE Robotics and Automation Letters*, 6(3):4782–4789, 2021.
- [68] Chul-Hong Kim and Dong-Il Cho. Dnn-based slip ratio estimator for lugged-wheel robot localization in rough deformable terrains. *IEEE Access*, 11:53468–53484, 2023.
- [69] Deok-Hwa Kim, Seung-Beom Han, and Jong-Hwan Kim. Visual odometry algorithm using an RGB-D sensor and IMU in a highly dynamic environment. In *Robot Intelligence Technology and Applications 3*, pages 11–26. Springer, 2015.
- [70] Jaesung Kim and Byung Kook Kim. Cornering trajectory planning avoiding slip for differential-wheeled mobile robots. *IEEE Transactions on Industrial Electronics*, 67(8):6698–6708, 2020.
- [71] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *Proc. 3rd Int. Conf. on Learning Representations*, 2015.
- [72] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *Intelligent Vehicles Symposium*, pages 486–492. IEEE, 2010.

- [73] Xianglong Kong, Wenqi Wu, Lilian Zhang, and Yujie Wang. Tightly-coupled stereo visual-inertial navigation using point and line features. *Sensors*, 15(6):12816–12833, 2015.
- [74] Urara Kono, Hiroshi Fujimoto, and Yoichi Hori. Localization of wheeled mobile robots from slip ratio estimation with simple model. In *International Conference on Mechatronics (ICM)*, pages 1–6. IEEE, 2021.
- [75] F Laura and Pascal Monasse. A critical review of the trifocal tensor estimation. In *Image and Video Technology: 8th Pacific-Rim Symposium, PSIVT 2017, Wuhan, China, November 20-24, 2017, Revised Selected Papers*, volume 10749, page 337. Springer, 2018.
- [76] Woosik Lee, Kevin Eickenhoff, Yulin Yang, Patrick Geneva, and Guoquan Huang. Visual-inertial-wheel odometry with online calibration. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4559–4566, 2020.
- [77] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.
- [78] Ao Li, Jikai Wang, Meng Xu, and Zonghai Chen. DP-SLAM: A visual SLAM with moving probability towards dynamic environments. *Information Sciences*, 556:128–142, 2021.
- [79] Mingyang Li and Anastasios I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.
- [80] Mingyang Li, Hongsheng Yu, Xing Zheng, and Anastasios I. Mourikis. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 409–416, 2014.
- [81] Yuan Ping Li, Marcelo H Ang, and Wei Lin. Slip modelling, detection and control for redundantly actuated wheeled mobile robots. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 967–972, 2008.
- [82] Anbang Liang, Qingquan Li, Zhipeng Chen, Dejin Zhang, Jiasong Zhu, Jianwei Yu, and Xu Fang. Spherically optimized ransac aided by an imu for fisheye image matching. *Remote Sensing*, 13(10):2017, 2021.

- [83] Jinxu Liu, Wei Gao, and Zhanyi Hu. Visual-inertial odometry tightly coupled with wheel encoder adopting robust initialization and online extrinsic calibration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5391–5397, 2019.
- [84] Miao Liu, Girish Chowdhary, Bruno Castra da Silva, Shih-Yuan Liu, and Jonathan P. How. Gaussian processes for learning and control: A tutorial with examples. *IEEE Control Systems Magazine*, 38(5):53–86, 2018.
- [85] Wanli Liu, Zhixiong Li, Shuaishuai Sun, Munish Kumar Gupta, Haiping Du, Reza Malekian, Miguel Angel Sotelo, and Weihua Li. Design a novel target to improve positioning accuracy of autonomous vehicular navigation system in GPS denied environments. *IEEE Transactions in Industrial Informatics*, 17(11):7575–7588, 2021.
- [86] Giuseppe Loianno, Michael Watterson, and Vijay Kumar. Visual inertial odometry for quadrotors on SE(3). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1544–1551, 2016.
- [87] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.
- [88] Xudong Long, Weiwei Zhang, and Bo Zhao. PSPNet-SLAM: A semantic SLAM detect dynamic object by pyramid scene parsing network. *IEEE Access*, 8:214685–214695, 2020.
- [89] G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.
- [90] Lukas Luft, Tobias Schubert, Stergios I Roumeliotis, and Wolfram Burgard. Recursive decentralized localization for multi-robot systems with asynchronous pairwise communication. *The International Journal of Robotics Research*, 37(10):1152–1167, 2018.
- [91] Shujun Ma, Xinhui Bai, Yinglei Wang, and Rui Fang. Robust stereo visual-inertial odometry using nonlinear optimization. *Sensors*, 19(17):3747, 2019.
- [92] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

- [93] Agostino Martinelli. Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *Transactions on Robotics*, 28(1):44–60, 2011.
- [94] Agostino Martinelli. Closed-form solution of visual-inertial structure from motion. *International journal of computer vision*, 106(2):138–152, 2014.
- [95] Agostino Martinelli, Alexander Oliva, and Bernard Mourrain. Cooperative visual-inertial sensor fusion: The analytic solution. *IEEE Robotics and Automation Letters*, 4(2):453–460, 2019.
- [96] Igor V. Melnyk, Joel A. Hesch, and Stergios I. Roumeliotis. Cooperative vision-aided inertial navigation using overlapping views. In *International Conference on Robotics and Automation*, pages 936–943. IEEE, 2012.
- [97] Sherif AS Mohamed, Mohammad-Hashem Haghbayan, Tomi Westerlund, Jukka Heikkonen, Hannu Tenhunen, and Juha Plosila. A survey on odometry for autonomous navigation systems. *IEEE Access*, 7:97466–97486, 2019.
- [98] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fast-SLAM: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598, 2002.
- [99] Lia Morra, Andrea Biondo, Nicola Poerio, and Fabrizio Lamberti. Mixo: Mixture of experts-based visual odometry for multicamera autonomous systems. *IEEE Transactions on Consumer Electronics*, 69(3):261–270, 2023.
- [100] Anastasios I Mourikis and Stergios I Roumeliotis. Performance analysis of multirobot cooperative localization. *Transactions on robotics*, 22(4):666–681, 2006.
- [101] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007.
- [102] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *Transactions on Robotics*, 31(5):1147–1163, 2015.
- [103] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGBD cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

- [104] Sylvie Naudet-Collette, Kathia Melbouci, Vincent Gay-Bellile, Omar Ait-Aider, and Michel Dhome. Constrained RGBD-SLAM. *Robotica*, 39(2):277–290, 2021.
- [105] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *International conference on computer vision*, pages 2320–2327. IEEE, 2011.
- [106] Barza Nisar, Philipp Foehn, Davide Falanga, and Davide Scaramuzza. Vimo: Simultaneous visual inertial model-based odometry and force estimation. *IEEE Robotics and Automation Letters*, 4(3):2785–2792, 2019.
- [107] Benjamin Noack, Joris Sijs, Marc Reinhardt, and Uwe D Hanebeck. Decentralized data fusion with inverse covariance intersection. *Automatica*, 79:35–41, 2017.
- [108] Diego Ortin and José Maria Martinez Montiel. Indoor robot motion based on monocular images. *Robotica*, 19(3):331–342, 2001.
- [109] Liam Paull, Guoquan Huang, Mae Seto, and John J Leonard. Communication-constrained multi-aur cooperative SLAM. In *International Conference on Robotics and Automation (ICRA)*, pages 509–516. IEEE, 2015.
- [110] Taihú Pire, Thomas Fischer, Gastón Castro, Pablo De Cristóforis, Javier Civera, and Julio Jacobo Berlles. S-ptam: Stereo parallel tracking and mapping. *Robotics and Autonomous Systems*, 93:27–42, 2017.
- [111] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.
- [112] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Tr. Robotics*, 34(4):1004–1020, 2018.
- [113] Tong Qin and Shaojie Shen. Robust initialization of monocular visual-inertial estimation on aerial robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4225–4232, 2017.
- [114] Tong Qin and Shaojie Shen. Online temporal calibration for monocular visual-inertial systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3662–3669. IEEE, 2018.

- [115] Mei Qu and Haijian Lu. Cooperative simultaneous localization and mapping with local measurements in 2d space. In *6th International Conference on Automation, Control and Robotics Engineering (CACRE)*, pages 509–513, 2021.
- [116] Long Quan. Invariants of six points and projective reconstruction from three uncalibrated images. *Transactions on Pattern Analysis and Machine Intelligence*, 17(1):34–46, 1995.
- [117] Rahul Raguram, Jan-Michael Frahm, and Marc Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision*, pages 500–513. Springer, 2008.
- [118] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.
- [119] Niraj Reginald, Omar Al-Buraiki, Baris Fidan, and Ehsan Hashemi. Confidence estimator design for dynamic feature point removal in robot visual-inertial odometry. In *48th Annual Conf. of the IEEE Industrial Electronics Society*, pages 1–6, 2022.
- [120] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.
- [121] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.
- [122] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to sift or surf. In *International Conference on Computer Vision*, pages 2564–2571, 2011.
- [123] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.
- [124] Matthias Schorghuber, Daniel Steininger, Yohann Cabon, Martin Humenberger, and Margrit Gelautz. Slamantic-leveraging semantics to improve VSLAM in dynamic environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

- [125] Soroush Sheikhpour and Mohamed M Atia. An enhanced visual-inertial navigation system based on multi-state constraint Kalman filter. In *IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 361–364. IEEE, 2020.
- [126] Soroush Sheikhpour and Mohamed Maher Atia. Calibration-free visual-inertial fusion with deep convolutional recurrent neural networks. In *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+)*, pages 2198–2209, 2019.
- [127] Hanmin Sheng, Jian Xiao, Yuhua Cheng, Qiang Ni, and Song Wang. Short-term solar power forecasting based on weighted gaussian process regression. *IEEE Transactions on Industrial Electronics*, 65(1):300–308, 2018.
- [128] Paloma Sodhi, Sanjiban Choudhury, Joshua G Mangelson, and Michael Kaess. ICS: Incremental constrained smoothing for state estimation. In *International Conference on Robotics and Automation (ICRA)*, pages 279–285. IEEE, 2020.
- [129] Ariane Spaenlehauer, Vincent Frémont, Y Ahmet Şekercioglu, and Isabelle Fantoni. A loosely-coupled approach for metric scale estimation in monocular vision-inertial systems. In *International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 137–143. IEEE, 2017.
- [130] Cyrill Stachniss, John J Leonard, and Sebastian Thrun. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, pages 1153–1176. Springer, 2016.
- [131] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.
- [132] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):1–11, 2017.
- [133] Tim Y. Tang, David J. Yoon, Francois Pomerleau, and Timothy D. Barfoot. Learning a bias correction for lidar-only motion estimation. In *15th Conference on Computer and Robot Vision (CRV)*, pages 166–173, 2018.
- [134] Sebastian Thrun. Simultaneous localization and mapping. In *Robotics and cognitive approaches to spatial mapping*, pages 13–41. Springer, 2007.

- [135] Philip HS Torr and Andrew Zisserman. Robust parameterization and computation of the trifocal tensor. *Image and vision Computing*, 15(8):591–605, 1997.
- [136] Nikolas Trawny, Xun S Zhou, Ke Zhou, and Stergios I Roumeliotis. Interrobot transformations in 3-d. *Transactions on Robotics*, 26(2):226–243, 2010.
- [137] Bill Triggs. Matching constraints and the joint image. In *Proc. of International Conference on Computer Vision*, pages 338–343. IEEE, 1995.
- [138] Filip Tronarp, Toni Karvonen, and Simo Särkkä. Mixture representation of the Matern class with applications in state space approximations and Bayesian quadrature. In *IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2018.
- [139] Florian Tschopp, Michael Riner, Marius Fehr, Lukas Bernreiter, Fadri Furrer, Tonci Novkovic, Andreas Pfunder, Cesar Cadena, Roland Siegwart, and Juan Nieto. Versavis—an open versatile multi-camera visual-inertial sensor suite. *Sensors*, 20(5):1439, 2020.
- [140] Evgeny Tsykunov, Valery Ilin, Stepan Perminov, Aleksey Fedoseev, and Elvira Zainulina. Coupling of localization and depth data for mapping using intel realsense t265 and d435i cameras. *arXiv preprint arXiv:2004.00269*, 2020.
- [141] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Computer Architecture Letters*, 13(04):376–380, 1991.
- [142] Kai Wang, Yimin Lin, Luowei Wang, Liming Han, Minjie Hua, Xiang Wang, Shiguo Lian, and Bill Huang. A unified framework for mutual improvement of SLAM and semantic segmentation. In *International Conference on Robotics and Automation (ICRA)*, pages 5224–5230. IEEE, 2019.
- [143] Runzhi Wang, Wenhui Wan, Yongkang Wang, and Kaichang Di. A new RGB-D SLAM method with moving object detection for dynamic indoor scenes. *Remote Sensing*, 11(10):1143, 2019.
- [144] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, 2017.

- [145] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In Arthur Gretton and Christian C. Robert, editors, *Proc. 19th Int. Conf. on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain, 09–11 May 2016.
- [146] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *CVPR 2011*, pages 3057–3064. IEEE, 2011.
- [147] 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. In *Robotics: Science and Systems*, volume 2, 2015.
- [148] Kejian J Wu, Chao X Guo, Georgios Georgiou, and Stergios I Roumeliotis. VINS on wheels. In *International Conference on Robotics and Automation (ICRA)*, pages 5155–5162. IEEE, 2017.
- [149] Kejian J Wu and Stergios I Roumeliotis. Unobservable directions of VINS under special motions. *Department of Computer Science & Engineering, University of Minnesota: Minneapolis, MN, USA*, 2016.
- [150] Zhuoyue Wu, Guirong Zhuo, and Feng Xue. Self-supervised monocular depth estimation scale recovery using ransac outlier removal. In *4th CAA International Conference on Vehicular Control and Intelligence*, pages 97–102. IEEE, 2020.
- [151] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 197–206, 2007.
- [152] Yulin Yang, Patrick Geneva, Kevin Ekenhoff, and Guoquan Huang. Degenerate motion analysis for aided ins with online spatial and temporal sensor calibration. *Robotics and Automation Letters*, 4(2):2070–2077, 2019.
- [153] Yulin Yang and Guoquan Huang. Observability analysis of aided INS with heterogeneous features of points, lines, and planes. *IEEE Transactions on Robotics*, 35(6):1399–1418, 2019.
- [154] Yulin Yang, James Maley, and Guoquan Huang. Null-space-based marginalization: Analysis and algorithm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6749–6755. IEEE, 2017.

- [155] Bharadwaja Yathirajam, Vaitheeswaran Sevoor Meenakshisundaram, and Ananda Challaghatta Muniyappa. An efficient approach to initialization of visual-inertial navigation system using closed-form solution for autonomous robots. *Journal of Intelligent & Robotic Systems*, 101(3):1–21, 2021.
- [156] Ying Yu, Kin Wong, M Chang, and Siu Or. Recursive camera-motion estimation with the trifocal tensor. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, 36(5):1081–1090, 2006.
- [157] Lilian Zhang. *Line primitives and their applications in geometric computer vision*. PhD thesis, University of Kiel, 2013.
- [158] Mingming Zhang, Xingxing Zuo, Yiming Chen, Yong Liu, and Mingyang Li. Pose estimation for ground robots: On manifold representation, integration, reparameterization, and optimization. *IEEE Transactions on Robotics*, 37(4):1081–1099, 2021.
- [159] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018.
- [160] Xinyang Zhao, Qinghua Li, Changhong Wang, Hexuan Dou, and Bo Liu. Robust depth-aided visual-inertial-wheel odometry for mobile robots. *IEEE Transactions on Industrial Electronics*, pages 1–11, 2023.
- [161] Jie Zhu, Yujie Tang, Xin Shao, and Yangmin Xie. Multisensor fusion using fuzzy inference system for a visual-imu-wheel odometry. *IEEE Transactions on Instrumentation and Measurement*, 70:1–16, 2021.
- [162] Jinlin Zhu, Muyun Jiang, Guohao Peng, Le Yao, and Zhiqiang Ge. Scalable soft sensor for nonlinear industrial big data via bagging stochastic variational gaussian processes. *IEEE Transactions on Industrial Electronics*, 68(8):7594–7602, 2021.
- [163] Pengxiang Zhu, Yulin Yang, Wei Ren, and Guoquan Huang. Cooperative visual-inertial odometry. In *Proc. International Conference on Robotics and Automation, Xi'an, China*, 2021.