On the Enhancement of the Localization of Autonomous Mobile Platforms

by

Mostafa Osman

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Science
in
Mechanical & Mechatronics Engineering

Waterloo, Ontario, Canada, 2020

Author's Declaration

Statement of Contributions

Chapter 1, 2 and 7 were authored solely by the author of the thesis. The co-authors offered supervision and revised some of the contents included in Chapter 2.

The idea presented in Chapter 3 was partially conceived by the author of the thesis. Additionally, the author partially contributed to the design of the proposed pipeline. Furthermore, the development of the RGB-D and stereo vision visual odometry algorithms as well as the testing and validation were executed by the author. Finally, the author wrote the first draft of the chapter, then the manuscript was revised by his co-authors.

The idea of the algorithm presented in Chapter 4 was conceived by the author of the thesis. The author also derived the drift error model of odometries and greatly contributed to the design and implementation of the algorithm. The author executed the validation experiments reported in this thesis. Finally, the author wrote the first draft of the chapter, which was then revised by his co-authors.

The idea of the localization based on moving horizon estimation method, presented in Chapter 5, was conceived by the author and was inspired by the work of T. Moore and D. Stouch in [98], who implemented a similar idea but using unscented and extended Kalman filters. Furthermore, the author implemented and validated the proposed scheme. Finally, the author of the thesis wrote the first draft of the chapter, which was revised by his co-authors.

The model predictive control proposed in Chapter 6 was implemented by the author of the thesis. The mobile manipulator modeling as well as the problem formulation of the model predictive control were conceived by the author. Furthermore, the author designed the simulation platform and the different scenarios used for validating the proposed controller. Finally, the author of the thesis wrote the first draft of the chapter, which was then revised by his co-authors.

Abstract

The focus of many industrial and research entities on achieving full robotic autonomy increased in the past few years. In order to achieve full robotic autonomy, a fundamental problem is the localization, which is the ability of a mobile platform to determine its position and orientation in the environment. In this thesis, several problems related to the localization of autonomous platforms are addressed, namely, visual odometry accuracy and robustness; uncertainty estimation in odometries; and accurate multi-sensor fusion-based localization. Beside localization, the control of mobile manipulators is also tackled in this thesis. First, a generic image processing pipeline is proposed which, when integrated with a feature-based Visual Odometry (VO), can enhance robustness, accuracy and reduce the accumulation of errors (drift) in the pose estimation. Afterwards, since odometries (e.g. wheel odometry, LiDAR odometry, or Visual Odometry (VO)) suffer from drift errors due to integration, and because such errors need to be quantified in order to achieve accurate localization through multi-sensor fusion schemes (e.g. extended or unscented kalman filters). A covariance estimation algorithm is proposed, which estimates the uncertainty of odometry measurements using another sensor which does not rely on integration. Furthermore, optimization-based multi-sensor fusion techniques are known to achieve better localization results compared to filtering techniques, but with higher computational cost. Consequently, an efficient and generic multi-sensor fusion scheme, based on Moving Horizon Estimation (MHE), is developed. The proposed multi-sensor fusion scheme: is capable of operating with any number of sensors; and considers different sensors measurements rates, missing measurements, and outliers. Moreover, the proposed multi-sensor scheme is based on a multi-threading architecture, in order to reduce its computational cost, making it more feasible for practical applications. Finally, the main purpose of achieving accurate localization is navigation. Hence, the last part of this thesis focuses on developing a stabilization controller of a 10-DOF mobile manipulator based on Model Predictive Control (MPC). All of the aforementioned works are validated using numerical simulations; real data from: EU Long-term Dataset, KITTI Dataset, TUM Dataset; and/or experimental sequences using an omni-directional mobile robot. The results show the efficacy and importance of each part of the proposed work.

Acknowledgements

First, I would like to thank my supervisors, Prof. Soo Jeon and Prof. William Melek, for giving me the opportunity to join their research group and their help and guidance through every step of this thesis work. Furthermore, I would like to thank Dr. Ahmed Hussein and Dr. Mohamed Mehrez, without them, I would not have managed to complete this thesis in its current form. I would also like to thank my amazing wife, Alya Wael, for being there for me during my study. Finally, and most importantly, I would like to thank my parents, who are my ultimate supporters and my main inspiration throughout life.

Dedication

This is dedicated to my father, Prof. Elsaid Osman, who is simply the most important person in my life.

Table of Contents

List of Figures

List of Tables

Abbreviations

AHE Adaptive Histogram Equalization , 27

AMCL Adaptive Monte Carlo Localization , 9, 19

AOR Angle based Outliers Rejection , 5, 26, 29–33, 38–41, 44–46, 48, 95

CLAHE Contrast Limited Adaptive Histogram Equalization , 5, 11, 26–29, 38, 40, 41, 43, 44, 48, 95

DCE Drift Covariance Estimation xiv, 5, 7, 49, 56, 58–64, 73, 95–97

EKF Extended Kalman Filter , 3, 13, 17, 18, 23

GPS Global Positioning System , 2, 3, 5, 35, 65, 66

IMU Inertial Measurement Unit , 2, 3, 10, 18–20, 35, 36, 65, 66, 72, 73

LOAM LiDAR Odometry and Mapping , 5

LSE Least Squares Estimation , 21, 22

MAP Maximum a Posteriori , 3, 4, 19, 20, 22, 64

MCL Monte Carlo Localization

MHE Moving Horizon Estimation iv, , 4, 6–8, 22–24, 64–68, 70, 72–81, 95, 96

MLE Maximum Likelihood Estimation , 19, 20

MM Mobile Mnaipulators

MPC Model Predictive Control iv, , 4, 6, 7, 24, 25, 82, 94

MR Mobile Robot

NLP Nonlinear Programming Problem , 88

NMPC Nonlinear Model Predictive Control , 83, 87–92, 96, 97

OCP Optimal Control Problem , 88, 89, 91, 96

ORB Oriented FAST and Rotated BRIEF , 10

PF Particle Filter

RANSAC RANdom SAmpling Consensus , 12

ROS Robot Operating System , 6, 33, 35, 72, 83

SFM Structure from Motion , 10

SIFT Scale Invariant Features Transform , 10

SLAM Simultaneous Localization and Mapping , 11, 14, 17, 18, 20

SSC Suppression via Square Covering , 5, 11, 12, 26, 29, 30, 32, 40, 41, 43, 44, 48, 95

SURF Speeded-up Robust Features , 10, 29, 33

TV True Variance , 60, 61

UGV Unmanned Ground Vehicle

UKF Unscented Kalman Filter xv, 3, 5, 6, 13, 18, 19, 56, 58, 61, 63–65, 73–77, 79–81, 96

VO Visual Odometry iv, xi, 2, 4, 5, 8–13, 18, 26–28, 30, 33–39, 41–49, 95, 96

List of Symbols

$\mathbb{N}$ The set of natural number.

$\mathbb{R}$ The set of real number.

$\mathbb{R}_{\geq 0}$ The set of non-negative real numbers.

t Time $t \in \mathbb{R}_{\geq 0}$.

τ Sampling time of a discrete time system where $\tau \in \mathbb{R}_{\geq 0}/\{0\}$.

n Number of states specified in context.

k The time-step index of a discrete time system where $t = k\tau$ and $k \in \mathbb{N}$.

$\mathbf{x}^a$ A state vector of a robotic arm $\mathbf{x}^a \in \mathbb{R}^n$.

$\mathbf{x}^b$ A state vector of a ground mobile robot (base) $\mathbf{x}^b \in \mathbb{R}^n$.

$\mathbf{x}$ A state vector of an autonomous platform $\mathbf{x} \in \mathbb{R}^n$.

$\mathbf{u}$ A control vector of an autonomous platform $\mathbf{u} \in \mathbb{R}^p$.

$\mathbf{z}$ A measurement vector of an autonomous platform $\mathbf{z} \in \mathbb{R}^m$.

$p(A|B)$ A probability density function of A given a condition B.

$(\cdot)_{k_1:k_2}$ The subscript represents a time-interval from time-step k_1 to time-step k_2, e.g., $\mathbf{x}_{1:k} := \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$.

$\mathbf{f}(\cdot)$ A process model mapping.

$\mathbf{h}(\cdot)$ A measurement model mapping.

$\mathbf{w}$ A random process noise vector.

$\mathbf{v}$ A random measurement noise vector.

F The process model Jacobian matrix.

H The measurement model Jacobian matrix.

$\hat{\mathbf{x}}$ An estimated state vector of an autonomous platform $\hat{\mathbf{x}} \in \mathbb{R}^n$.

$\hat{\mathbf{z}}$ An estimated measurement vector of an autonomous platform $\mathbf{z} := \mathbf{h}(\hat{\mathbf{x}}) \in \mathbb{R}^m$.

X The feasible states set $X \subseteq \mathbb{R}^n$.

U The feasible controls set $U \subseteq \mathbb{R}^p$.

Z The feasible measurements set $Z \subseteq \mathbb{R}^m$.

$\mathbf{X}$ The state sequence for the whole state history $\mathbf{X} := (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k) \in X^{k+1}$.

$\mathbf{Z}$ The measurements sequence for the whole state history $\mathbf{Z} := (\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k) \in Z^k$.

$\mathcal{Q}$ The process noise covariance.

$\mathcal{R}$ The measurement noise covariance.

$\mathbf{P}$ The Posterior covariance $\mathbf{P} \in \mathbb{R}^{n\times n}$

$\mathbf{e}_\mathbf{x}$ The error function for the state vector $\mathbf{e}_\mathbf{x} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$.

$\mathbf{e}_\mathbf{z}$ The error function for the measurement vectors $\mathbf{e}_\mathbf{z} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$.

$||\cdot||_2$ The l_2-norm operator.

$||\cdot||_1$ The l_1-norm operator.

$||\mathbf{d}||_A$ The weighted l_2-norm of a vector $||\mathbf{d}||_A := \sqrt{\mathbf{d}^\top A \mathbf{d}}$.

$\mathbf{p}$ A position vector of the autonomous platform $\mathbf{p} := \begin{bmatrix} x & y & z \end{bmatrix}^\top$.

θ An orientation vector of the autonomous platform in some representation specified in context.

$\mathbf{I}_{c_1\times c_2}$ An $c_1 \times c_2$ Identity matrix.

C A measurement (output) matrix of a sensor.

$\mathbf{0}_{c_1 \times c_2}$ An $c_1 \times c_2$ all zeros matrix.

$\hat{\cdot}$ The cap indicate a noisy measurement or an estimate, e.g., $\hat{\mathbf{x}}$ is the state estimate.

$SE(3)$ The special Euclidean group.

$SO(3)$ The special orthogonal group.

R A rotation matrix $R \in SO(3)$.

N The estimation horizon of MHE.

N_c The prediction horizon of MPC.

$||\mathbf{d}||_\infty$ The l_∞ norm defined as $||\mathbf{d}||_\infty := \max_{i \in [1:\text{dim}]} |d_i|$

Tr The matrix trace operator.

$[A]_i$ The i-th column vector of a matrix A.

$\mathcal{X}_N$ The state sequence over the prediction horizon of an MPC.

$\mathcal{U}_N$ The control sequence over the prediction horizon of an MPC.

$\mathcal{N}(\mu, \Sigma)$ A Gaussian distribution with the mean μ and the covariance Σ.

Chapter 1

Introduction

In this chapter, the motivation of this thesis; the problems addressed; and the proposed solutions are introduced. The thesis objectives and contributions are also stated. Finally, the thesis structure and organization are described.

1.1 Motivation

Robotics underwent a noticeable evolution during the past few years and is now an essential part of a large number of industrial processes. Currently, one of the main objectives of many industries is to introduce solutions allowing increased levels of autonomy in their processes. Autonomous robotics can help with many tasks, which are still conducted by human workers or a robotic system with human supervision/operation, such as warehouse management [74], mines search and detection [1], farming [52], underwater exploration [166], and so on. Using a fully autonomous robotic agent may lead to better efficiency, accuracy, and human safety. By the same token, automotive industry also achieved great advances in autonomy, by developing the driver assistance systems such as adaptive cruise control [153, 96], collision avoidance [135, 46] and automatic parking [137]. Building fully autonomous self-driving vehicles is a priority, because it will lead to a large reduction in the amount of traffic accidents through eliminating its main cause; the human error (Distracted driving, speeding, drunk driving, and reckless driving) [27].

1.2 Problem Statement

Accurate localization is essential to achieve full robotic autonomy. Localization is the ability of a robotic platform to determine its position and orientation in the environment [138]. It is essential for an autonomous platform to interact with the environment to, for example, navigate or perform manipulation tasks. Consequently, achieving accurate, and robust localization is one of the essential milestones to achieve full autonomy. Although the localization of a robotic platform using off-board sensors, such as cameras [45] or adding infrastructure to the environment, such as magnetic tape [125], visual landmarks [57], wi-fi hot-spots [51] might be simple, the more general problem of localization using on-board sensors without additional infrastructure is challenging, mainly due to the physical limitations of the sensors used.

In this thesis, several unresolved issues related to robot localization are addressed. These are:

- Visual Odometry (VO) accuracy and robustness,
- uncertainty estimation in odometries, and
- accurate multi-sensor fusion-based localization.

1.2.1 Visual Odometry Accuracy and Robustness

Currently, several sensors are used to achieve localization including LiDAR [175], Radar [119], Global Positioning System (GPS) [35], Inertial Measurement Unit (IMU) [173], wheel encoders [77], and cameras [136].

One of the common methods for localization using cameras is VO [133][47]. VO estimates the ego-motion of a camera by determining the incremental motion between the successive camera frames. Like other odometry methods (wheel encoders, LiDAR, ... and so on), VO integrates the incremental motion between successive frames to compute the overall trajectory of the camera, leading to drift errors over long distances.

VO algorithms have a drawback since they rely on integration, which may suffer from drift errors due to, for example, false matched features, bad lighting and illumination problems, random noise or motion bias [170].

1.2.2 Uncertainty Estimation in Odometries

To achieve accurate localization, the use of highly accurate sensors such as differential Global Positioning System (GPS) or motion capture systems is needed. However, the use of these systems can be very costly. Another way of achieving accurate localization is the use of multi-sensor fusion techniques with sensors of lower accuracy and cost. This in turn can lead to an accurate estimate of the pose of the platform (position and orientation) [152, 63, 88, 36, 90, 87].

In order to accurately estimate the pose of an autonomous platform by multi-sensor fusion, the statistics of the noise affecting each sensor must be known. Almost all multi-sensor fusion algorithms uses the covariance matrix during operation. This covariance can be in its direct form as in the Kalman filters, as the importance factor in the particle filter, or as the information matrix (the inverse of the covariance matrix) in the information filter and optimization-based techniques [147].

Unfortunately, determining the covariance matrices is not straightforward and, in most cases, infeasible. To determine the covariance matrix of a sensor, the ground-truth is needed, which is generally very difficult to acquire. Furthermore, even if the ground-truth is available, the localization of a mobile platform depends on odometries (encoders odometry, visual odometry, IMU odometry, and so on), which all suffer from accumulation of error (as discussed earlier). This error accumulation has a random nature, which depends on the environmental conditions as well as the type of sensor. Accordingly, this causes an increase in the covariance values during operation [138, 130].

1.2.3 Accurate Multi-sensor Fusion-based Localization

Multi-sensor fusion-based localization can be achieved through two approaches; probabilistic (filtering) and optimization-based. probabilistic methods rely on the Bayesian estimation theory (e.g., Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) [60, 59, 4] or particle filters [147]). All of these techniques rely on a motion model for computing the prior distribution of the pose while using the sensors measurements to recursively estimate the posterior probability of the pose.

Probabilistic methods are recursive and rely on the latest estimate of the state as well as the latest control action and measurement in order to estimate the current state. On the other hand, Optimization based techniques are based on Maximum a Posteriori (MAP) estimation, which aims at maximizing the posterior distribution over the whole estimated trajectory; this is achieved through solving a least squares optimization problem over the innovations (measurements error) of different sensors' measurements [22].

Utilizing the full information through the trajectory leads to the optimization-based methods achieving better accuracy compared to filtering techniques. However, the main drawback of a full information MAP is intractability over long duration of operation, due to the increase of optimization variables (unknown poses in case of localization). In practice, the effect of older states over the current state decreases with time. Therefore, optimizing over the whole history of states may become unnecessary and computationally expensive.

1.3 Thesis Contributions

This thesis proposes methods to enhance the localization of autonomous platforms through addressing the problems discussed in Section 1.2. To this end, the objectives of this thesis are as follows:

1. Developing a filtration pipeline, which can be integrated to any Visual Odometry (VO) algorithm, to enhance the robustness and accuracy of localization. Such pipeline should be generic and can be integrated to any type of feature-based VO (monocular, stereo, or RGB-D).

2. Estimating the drift errors statistics in the odometry output during the operation of the mobile platform. The estimation algorithm should be generic, in the sense that it can be used with any type of odometry (encoders, visual, LiDAR, …etc.). Such algorithm can be used with any multi-sensor fusion (filtering or optimization-based) algorithm to achieve accurate localization.

3. Develop an optimization-based multi-sensor fusion scheme using Moving Horizon Estimation (MHE) for localization. Such scheme should be designed to use any number of on-board sensors, different sensors rates, missing measurements, and outlier rejection. Furthermore, one of the main drawbacks of optimization-based multi-sensor fusion schemes is its computational cost, hence, the localization scheme must be computationally efficient to be useful for practical implementations.

4. Finally, design a point-stabilization Model Predictive Control (MPC) of a whole-body mobile manipulator for autonomous navigation and manipulation. The proposed controller can be integrated with the proposed localization algorithms to achieve better autonomous operation.

The solutions proposed for the mentioned problems in Section 1.2 are the following:

1.3.1 A generic image processing pipeline for enhancing the visual odometry accuracy and robustness

A generic and modular image processing pipeline is proposed to enhance the accuracy and robustness of feature-based VO algorithms, which can be applied to any type of VO; monocular, RGB-D or stereo vision.

The proposed pipeline includes additional filtration and pre-processing stages through Contrast Limited Adaptive Histogram Equalization (CLAHE) filter with adaptive thresholding to overcome lighting changes [124]. Additionally, the Suppression via Square Covering (SSC) is used to avoid any bias in the motion estimation [12]. Finally, a novel outlier rejection algorithm, referred to as the Angle based Outliers Rejection (AOR), is proposed, to reject false matched features as well as features captured on a moving object in the scene. The pipeline is integrated into a monocular, RGB-D, and stereo VO and validated using KITTI [49] and TUM datasets [144], as well as experimental sequences generated by an omni-directional mobile robot.

1.3.2 An Online Approach for Estimating the Covariance of Drift in Odometries

The drift error model for any type of odometry and the drift covariance formula are derived. Afterwards, the estimation methodology introduced in [110] is generalized and more formally presented as the DCE algorithm. The estimation methodology can enhance the accuracy of localization, when integrated to odometry algorithms. Moreover, the proposed DCE is an online algorithm, i.e. it can estimate the covariance of an odometry while the mobile platform is operating.

For validating the proposed approach, a comparative study is conducted using several sequences from the EU Long-term Dataset [169]. The DCE is applied to the LiDAR Odometry and Mapping (LOAM) measurement [175] which are then fed to a UKF multi-sensor fusion scheme alongside with noisy GPS measurements. The effect of the proposed algorithm on enhancing multi-sensor fusion localization accuracy is studied by comparing the localization output to that resulting from using several constant covariances for the LOAM measurements.

1.3.3 A Generic Multi-sensor Fusion Scheme for Localization of Autonomous Platforms Using Moving Horizon Estimation

A generic multi-sensor fusion framework is developed for the localization of autonomous platforms using Moving Horizon Estimation (MHE). The proposed method is based on a multi-threading architecture for efficient computation in practical applications. The MHE fusion scheme is tested using both simulated data as well as experimental data sequences. The MHE estimation output is compared to that of a UKF. Finally, an implementation of the proposed estimation scheme is made open-source for the benefit of the scientific community. The main contributions of the proposed localization technique are:

- The development of a generic multi-sensor fusion scheme based on MHE for the localization of autonomous systems using any number of on-board sensors. The generic fusion scheme considers different sensor rates as well as missing measurements and outliers.
- A threaded realization of the MHE algorithm to reduce its computational cost to become more feasible for practical applications.
- A ROS-based open-source implementation of the proposed methodology to facilitate its validation and ease of deployment on experimental studies.

1.3.4 End-effector Stabilization of a 10-DOF Mobile Manipulator Using Nonlinear Model Predictive Control

A nonlinear Model Predictive Control (MPC) scheme is proposed to stabilize the end-effector of a 10-DOF mobile manipulator. First, the task-space kinematic model of the mobile manipulator is formulated, where the overall system rotations are expressed using the 3D special orthogonal group $SO(3)$. This model is used for state prediction in the nonlinear MPC formulation, which considers state, control, and kinematic singularity constraints. The proposed nonlinear MPC is implemented using Robot Operating System (ROS) and the efficacy of the proposed controller is demonstrated through a series of real-time simulations using Gazebo dynamic simulator [68]. The results show highly accurate and smooth stabilization of the end-effector.

1.4 Thesis Structure

This thesis is organized as follows:

- **Chapter 1** states the motivation behind the proposed work and the problems tackled in the thesis. Afterwards, the thesis objectives and contributions are stated. Finally, the overall structure of the thesis is described.
- **Chapter 2** discusses the related works to the thesis, which includes the odometry algorithms recently introduced to solve the localization problem. Moreover, the different probabilistic multi-sensor fusion approaches used to solve the localization problem are discussed. Finally, the last section discusses the literature on mobile manipulators control.
- **Chapter 3** considers the development of a generic image processing pipeline which enhances the performance of visual odometries.
- **Chapter 4** introduces the DCE algorithm which estimates the covariance of odometry measurements during the operation of the autonomous platform.
- **Chapter 5** discusses the generic MHE multi-sensor fusion-based localization scheme.
- **Chapter 6** discusses the kinematic modeling of a 10-DOF mobile manipulator and introduces the proposed point-stabilization MPC.
- Finally, **chapter 7** contains the final concluding remarks as well as the future works.

Chapter 2

Literature Review and Background[1]

2.1 Introduction

This chapter focuses on the previous work related to this thesis based on the proposed objectives. Section 2.2 includes:

- The latest breakthroughs in the development of VO algorithms.
- The research targeting the estimation of uncertainty in sensors used in localization.
- The main sensor fusion techniques used in localization.
- The origin of MHE as well as the recent work proposed for achieving localization using it.

Afterwards, the state of the art, in the control of mobile manipulators, is reviewed in Section 2.3.

2.2 Localization of Autonomous Mobile Robots

Localization is the task of estimating the position and orientation, referred to as pose, of a mobile platform. For autonomous operation, localization becomes crucial, since executing

[1]Some of the content of this chapter was published as Mostafa Osman, Ahmed Hussein, and Abdulla Al-Kaff, "Intelligent Vehicles Localization Approaches between Estimation and Information: A Review," in *Proc. IEEE Int. Conf. of Vehicular Electronics and Safety (ICVES)*, 2019 [109].

any task such as navigation, manipulation (for mobile manipulators), or exploration will not be possible without the determination of the pose. To this end, throughout the past decade, a large amount of research was conducted to achieve accurate localization.

2.2.1 Odometries

Historically, an odometer was a sensor which measures the velocity of a moving vehicle, as well as the distance traveled by it. Normally, this sensor would be an incremental or absolute encoder generating electrical pulses during the wheel rotation. By counting these pulses, the speed and distance traveled can be calculated. Lately, the scientific community started referring to any method, which integrates incremental motion, as an odometry. Different types of odometries were developed, such as LiDAR odometry [181, 26] and Visual Odometry (VO) [133, 47], which uses a LiDAR or a camera to determine the incremental motion between two successive instances. The overall path of the vehicle can then be determined by integrating these successive incremental motions.

Obviously, the sensors used in developing odometries are not ideal and are susceptible to noise. By relying on integration, the estimated pose by the odometry suffers from an accumulation of errors, namely, the drift or the drift error. To this end, many researchers developed different types of odometries which aim at reducing the drift [64, 134, 65].

Dead-reckoning (also referred to as encoder odometry) is the original method for determining the pose of ground mobile robots. Using the measurements from encoders fixed on the wheels of the robot, and through the use of the kinematic model of the robot, the pose is determined. Dead-reckoning was used extensively for the localization of mobile robots, however, using only dead-reckoning can lead to very inaccurate results or even complete divergence. Consequently, additional sensors were integrated to the system to correct for the drift. In [156], the dead-reckoning measurements were fused with a gyroscope and a magnetometer in order to achieve more accurate localization. In [150], the measurements from two encoders, a gyroscope and a magnetometer were fused with the measurements from ultrasonic sensors, in order to correct the drift error.

Although dead-reckoning is still used in several localization algorithms such as Adaptive Monte Carlo Localization (AMCL) [147, ch. 8] , the use of visual and LiDAR odometries is now more common. Unlike dead-reckoning, which relies on proprioceptive sensors [138] for determining the motion of the platform, LiDAR and cameras are exteroceptive sensors, which measure the change in the environment instead of the change in the platform itself. The use of LiDAR and cameras in odometries is based on a very similar concept, which is computing the transformation between the environment features in two consecutive instances of time [105, 28]. This transformation describes the motion of the mobile platform (assuming a static environment). Hence, the overall motion of the platform can be determined through concatenating the incremental transformations estimated by the odometry.

2.2.2 Visual Odometry

VO can be classified by the type of camera used in the motion estimation to monocular, stereo and RGB-D (red-green-blue-depth). Alternately, it can also be classified using the method of motion estimation into feature-based and direct.

Monocular VO uses images captured by only one camera to determine its trajectory. This technique usually relies on Structure from Motion (SFM) [149, 48, 53, 131]. With one camera, the motion of the robot can be captured up to an unobservable scale. This scale can then be determined through the use of an external velocity measurement from a wheel encoder or an IMU. Several researches also developed methods for estimating the scale of monocular VO without the use of external sensors such as [183, 182]. Lately, researchers have also started developing deep-learning techniques for monocular VO [79, 174].

Unlike the monocular VO, the stereo and the RGB-D VO estimate the full pose of the vehicle without any external measurements [143, 144, 55, 81].

All three categories of VO can be either direct or feature-based approaches. On the one hand, the feature-based method relies on visual features for calculating the transformation between consecutive frames. For the detection of such features, a feature detector and descriptor is used such as Scale Invariant Features Transform (SIFT) [86], Speeded-up Robust Features (SURF) [14], or Oriented FAST and Rotated BRIEF (ORB) [129] (see [15] for a comparison between the different detectors and descriptors). On the other hand, direct approaches compute the relative transformation between frames, based on the whole image intensities [7, 40].

Although all these VO techniques are well designed, VO algorithms have a drawback that it relies on integration, which may suffer from drift errors due to, for example, false matched features, bad lighting and illumination problems, random noise or motion bias [170].

Image Filtration

Several works addressed the problem of noise in images for VO enhancement. The noise may be attributed to poor lighting conditions caused by light source flare, random visual sensor noise, or other noise sources [120].

In [25], a direct VO algorithm using binary descriptors was used to overcome poor lighting conditions. The authors showed that the algorithm performed in a robust and efficient way even under low lighting conditions. This was accomplished by the illumination invariance property of the binary descriptor within a direct alignment framework. The VO algorithm proposed therein is a direct method, which is usually more computationally expensive compared to feature-based VO.

In [180], a robust feature-matching scheme was combined with an effective anti-blurring frame. The algorithm uses the singular value decomposition to mitigate the effect of blurring due to vibrations or other factors.

In [171], a stereo visual Simultaneous Localization and Mapping (SLAM) algorithm was proposed, which uses Contrast Limited Adaptive Histogram Equalization (CLAHE) filter to locally enhance the image contrast and get more feature details. The CLAHE-enhanced SLAM algorithm was compared to the results of a VO enhanced by a conventional histogram equalization and the results of ORB-SLAM2 [101]. The results showed a superior performance of the CLAHE-enhanced algorithm compared to the other algorithms. Furthermore, in [178], a robust VO for underwater environment was proposed. In order to overcome the turbid image quality of underwater imaging, the authors used CLAHE for contrast enhancement. The authors showed that the use of CLAHE resulted in brighter and larger visible regions. As a result, unclear structures were made more clear.

Non-maximal Suppression

Non-maximal suppression can be used to avoid poor distribution of features over the image, which leads to poor VO performance and motion bias. Several non-maximal suppression algorithms were used in VO [168], [58]. In [99], a feature descriptor was proposed to facilitate fast feature matching processing while preserving matching reliability. The authors chose to use the FAST (Features from Accelerated Segment Test) detector [127] along with a non-maximal suppression algorithm.

In [5], a stereo/RGB-D VO was proposed for mobile robots. Therein, the authors used the adaptive non-maximal suppression introduced in [19] to enhance the performance of the feature detector algorithm BRIEF (Binary Robust Independent Elementary Features) [23] by ensuring uniform distribution of features over the image.

In [12], three new and efficient adaptive non-maximal suppression approaches were introduced, which included the Suppression via Square Covering (SSC) algorithm. The positive impact of the three algorithms on visual SLAM was demonstrated. Authors in [12] showed that the output of the three algorithms is visually and statistically similar, however, SSC showed lower computational cost which suggests that it is more suitable for real-time applications such as VO.

Outliers Rejection

Feature-based VO relies on feature detecting and matching for motion estimation. Commonly, feature matching algorithms generate a considerable amount of false matched features [133]. These false matched features lead to the increased error in motion estimation or the complete divergence of the VO output. Several works in the literature addressed this problem. In [20], an iterative outlier rejection scheme for stereo-based VO was proposed. The proposed algorithm was designed to improve the VO motion estimation for high-speed and large-scale depth environments.

In [42], a stereo VO was proposed which relies on using reference frames instead of all frames. This was accomplished through first selecting the stable features from a frame using the quad-matching testing and the grid-based motion statistics. Afterwards, the features in this frame were matched to the features in a reference frame (instead of the previous frame), which contains the stable features found in the current frame.

A commonly used outlier rejection approach is the RANdom SAmpling Consensus (RANSAC). RANSAC is an iterative outlier rejection algorithm, which relies on the computation of model hypotheses, from randomly selected set of the matched features, followed by the verification of the hypotheses using the rest of the matched features [47]. In [67], a stereo VO algorithm was proposed which uses a RANSAC based outlier rejection along with an iterated sigma point Kalman filter to achieve robust frame-to-frame VO performance. Although RANSAC is effective in removing outliers, the iterative process sometimes results in poor performance due to a large number of iterations for convergence. Furthermore, if the number of outliers in the matched points is large, this may lead to wrong convergence entailing incorrect motion estimation.

2.2.3 Uncertainty Quantification

Ever since Rudolf Kalman proposed the Kalman filter in 1960 [62] and the continuous Kalman and Bucy filter in 1961 [61], the endeavors for quantifying the sensors covariances started [92]. Quantifying the noise covariance for a sensor in a multi-sensor fusion scheme was the subject of several works in the literature. The difficulty in quantifying such covariances resulted in the emergence of adaptive filtering techniques [34, 93].

In [172], a combination of a fuzzy logic controller and a conventional Kalman filter for an INS/GPS was proposed, to correct both the process noise covariance and the measurement noise covariance. The algorithm was validated using a simulation with an EKF, UKF and an iterated EKF. The drawback of this method is that, the design of the membership function for each sensor needed to estimate the covariance, may be challenging. Especially because each sensor differs in nature and can suffer from different noise sources. Additionally, in case of odometries, the behavior of drift is unpredictable and designing a membership function to describe it may be impractical.

Other works used adaptive Kalman filters to estimate the covariance matrices [3]. In [159], a Kalman filter with recursive estimation was presented; to estimate the noise covariance matrix of the measurements of linear time-invariant systems. Moreover, in [44], a stability analysis was performed to verify the stability of the estimator. However, this work was introduced to linear systems and assuming that the sensors are not suffering from drift.

In [123, 122], a method based on Bayesian Maximum Entropy and Interacting Multiple Model was proposed to detect and correct the drift in sensors used in Internet of things (IoT) technologies. In [148, 145], an online method to quantify and eliminate the random drift error in a Fiber Optic Gyro (FOG) was proposed through adaptive Kalman filtering. Such works only addressed the random drift noise in FOG and not in general odometries or general sensors suffering from random drift.

Beside the adaptive filtering approaches, systematic drift errors due to aging and mis-calibration were also studied. In [162], a blind calibration algorithm was proposed to calibrate the sensor drift using signal space projection and Kalman filtering. However, in such case, the drift being addressed is the systematic drift which can be eliminated through calibration. Moreover, in [161], a deep learning approach was proposed to address the same issue.

Similarly, most of the literature deals with drift as a systematic error which occurs due to the loss of calibration, temperature conditions, aging of the sensors or other environmental conditions. In such cases, this kind of error can be eliminated through calibration or compensation [21, 76, 16].

There are several works in the literature dealing with drift in odometry. However, in these works, the methodology adopted by the authors was based on reducing the error in the odometries. In [33], a VO algorithm, which uses a newly developed feature descriptor, was introduced to reduce the drift. Similarly, in [114], a convolution neural network was used to infer the sun direction which, in turn, was used to reduce the drift error in the VO.

In [78], a fusion scheme of monocular vision and radio-based ranging was introduced, to reduce the drift error in a SLAM algorithm. In [146], a learning approach was used to solve the drift problem in a LiDAR-only motion estimation. Although, all these works tried to reduce drift, it cannot be completely eliminated (except by ideal sensors). In reality, sensors used in odometries suffer from various types of errors due to ground conditions (slippage), temperature changes, driving behavior or lighting conditions. Moreover, due to the change of operational conditions, the drift itself cannot be predicted and the rate of accumulation of error changes with time. Therefore, it will be useful to estimate the amount of drift in a given odometry during operation (e.g. to achieve better multisensor fusion).

2.2.4 Multisensor Fusion Localization

Beside odometry, localization can also be tackled using the estimation theory techniques, by fusing the measurements from different sensors. This approach stems from the Bayesian estimation techniques.

Bayesian Estimation

Consider a discrete nonlinear system in the following form:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k), \tag{2.1}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k), \tag{2.2}$$

where $k \in \mathbb{N}$ is the time-step, $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{u} \in \mathbb{R}^p$ is the control vector, $\mathbf{z} \in \mathbb{R}^m$ is the measurement vector, $\mathbf{w} \in \mathbb{R}^q$ is the process noise, $\mathbf{v} \in \mathbb{R}^r$ is the measurement noise, $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^n$ is the nonlinear process model, and $\mathbf{h} : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^m$ is the nonlinear measurement model.

The state vector $\mathbf{x}$ and the measurement vector $\mathbf{z}$ are random processes, because they are subjected to both process and measurement noises, respectively. Consequently, one must use probabilities while dealing with such systems. In other words, the determination of the system (mobile platform) states becomes bounded to determining the probability of occurrence of the state given the process and measurement noises. This probability is called the posterior probability [147]

$$p(\mathbf{x}_k | \mathbf{z}_{1:k}, \mathbf{u}_{1:k}). \tag{2.3}$$

Assuming that the state $\mathbf{x}_k$ is complete, i.e., estimating $\mathbf{x}_{k+1}$ depends only on the state $\mathbf{x}_k$. In other words, the previous states $\mathbf{x}_{0:k-1}$, controls $\mathbf{u}_{1:k-1}$, and measurements $\mathbf{z}_{1:k-1}$ do not add any information to refine the estimation of the state vector $\mathbf{x}_{k+1}$. This assumption is referred to as the Markov assumption. Applying it to the posterior in (2.3) yields

$$p(\mathbf{x}_k|\mathbf{z}_{1:k},\mathbf{u}_{1:k}) = p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{z}_k,\mathbf{u}_k). \tag{2.4}$$

Applying the well-known Bayes rule to the posterior in Eq. (2.4) yields

$$p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{z}_k,\mathbf{u}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_{k-1:k},\mathbf{u}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)}{p(\mathbf{z}_k)}. \tag{2.5}$$

Observe the conditional independence between $\mathbf{z}_k$, and $\mathbf{x}_{k-1}$ and $\mathbf{u}_k$. This independence allows Eq. 2.5 to be written as

$$p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{z}_k,\mathbf{u}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)}{p(\mathbf{z}_k)}. \tag{2.6}$$

$p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)$ is commonly referred to as the prior probability. It is defined as the probability of occurrence of a state given the previous states and the current controls.

Using the law of total probability, the prior is calculated by integrating over the posterior of the previous state $\mathbf{x}_{k-1}$ jointly with the state transition probability (system model) distribution $\bar{p}$.

$$p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k) = \int \bar{p}(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)p(\mathbf{x}_{k-1}|\mathbf{x}_{k-2},\mathbf{z}_{k-1},\mathbf{u}_{k-1})dx_{k-1} \tag{2.7}$$

Using Eq. (2.6) and (2.7), the posterior of state $\mathbf{x}_k$ can be estimated using the state $\mathbf{x}_{k-1}$, the control $\mathbf{u}_k$ and the measurement $\mathbf{z}_k$. This estimation strategy is called the recursive Bayes filter (see Algorithm 1).

In [38], a multi-sensor fusion framework for sensors used in autonomous vehicles, based on the Bayesian filter, was proposed. In [103], a discrete Bayes filter was utilized to develop a lifelong localization algorithm of an autonomous mobile robot. In [18], a distributed Bayesian approach for data association and positioning was proposed in cooperative vehicular networks.

Bayes filter is the basis of probabilistic estimation. Through adding certain assumptions to the estimation problem, different filters can be derived based on the Bayes filter.

Algorithm 1: Bayes Filter

```
for $k \in \{1, ..., \infty\}$ do
    $$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) = \int \bar{p}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) p(\mathbf{x}_{k-1}|\mathbf{x}_{k-2}, \mathbf{z}_{k-1}, \mathbf{u}_{k-1}) dx_{k-1} \quad (2.8)$$
    $$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k, \mathbf{u}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)}{p(\mathbf{z}_k)} \quad (2.9)$$
end
```

Kalman Filter

The Kalman filter [62] is the most widely used estimator for linear systems. Kalman filter considers a linear time varying system

$$\mathbf{x}_k = A_k \mathbf{x}_{k-1} + B_k \mathbf{u}_k + \mathbf{w}_k \tag{2.10}$$

$$\mathbf{z}_k = C_k \mathbf{x}_k + \mathbf{v}_k \tag{2.11}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$, and $C \in \mathbb{R}^{m \times n}$ are the system, input and output matrices, respectively.

The Kalman filter assumes additive zero-mean Gaussian white noises, where white means that the noise at time-step k is independent from the noise at the previous time-steps.

$$\mathbf{w}_k \sim \mathcal{N}(0, \mathcal{Q}_k), \qquad \mathbf{v}_k \sim \mathcal{N}(0, \mathcal{R}_k), \tag{2.12}$$

where $\mathcal{Q}_k \in \mathbb{R}^{n \times n}$ and $\mathcal{R}_k \in \mathbb{R}^{m \times m}$ are the covariance matrices of the process and measurement noises at time-step k, respectively.

Several derivations of the famous Kalman filter, using different methods and perspectives, were developed. The Kalman filter can be derived using the recursive Minimum Mean Squared Error (MMSE) estimation as in [140, Ch. 5]. Moreover, the Bayes filter equations derived in the previous section can be used to drive it. For the rigorous derivation, see [147, Ch. 3].

The Kalman filter algorithm is stated in Algorithm 2, where $\mathbf{P}_0$ is the covariance of the initial state, $\hat{\mathbf{P}}_{k|k-1} \in \mathbb{R}^{n \times n}$ and $\hat{\mathbf{P}}_{k|k} \in \mathbb{R}^{n \times n}$ are the prior and posterior covariance matrices respectively, and $\mathcal{K}$ is the Kalman gain. Notice that $k|k-1$ indicates the prior, estimated using the information from the previous time-step (equivalent to Eq. 2.5 in the

```
Algorithm 2: Kalman Filter
  Initialize with: $\mathbf{P}_0$, $\mathbf{x}_0$
  for $k \in \{1, ..., \infty\}$ do
      Prediction Step:
          $\hat{\mathbf{x}}_{k|k-1} = A_k \hat{\mathbf{x}}_{k-1|k-1} + B_k \mathbf{u}_k$
          $\hat{\mathbf{P}}_{k|k-1} = A_k \mathbf{P}_{k-1|k-1} A_k^\top + \mathcal{Q}_k$
      Update Step:
          $\mathcal{K}_k = \hat{\mathbf{P}}_{k|k-1} C_k^\top (C_k \hat{\mathbf{P}}_{k|k-1} C_k^\top + \mathcal{R}_k)^{-1}$
          $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathcal{K}_k(\mathbf{z}_k - C_k \hat{\mathbf{x}}_{k|k-1})$
          $\hat{\mathbf{P}}_{k|k} = (I - \mathcal{K}_k C_k)\hat{\mathbf{P}}_{k|k-1}$
  end
  where the $\hat{\cdot}$ indicates an estimate (not the true value).
```

Bayes filter), and $k|k$ indicates the posterior estimated using the information from the current time-step (equivalent to Eq. 2.6 in the Bayes filter).

The Kalman filter algorithm is the **optimal** estimator, for linear time invariant systems, in a least squares sense (if and only if the previously stated assumptions are met).

Extended Kalman Filter

The direct extension to the Kalman filter to nonlinear systems is the Extended Kalman Filter (EKF). The EKF approximates the process and measurement models, using a first order Taylor expansion, to propagate the uncertainty. Needless to say, this approximation cancels the optimality of the Kalman filter. However, it can operate sub-optimally for quasi-linear systems (system with low nonlinearities). In case of highly nonlinear systems, the performance of the EKF deteriorates.

The EKF was extensively used throughout the past few years in both localization and filtering-based SLAM. In [41], an EKF-SLAM back-end, based on the detection of heterogeneous landmarks with a front-end based on visual data, was proposed. In [165], an integration between the path following and the SLAM problem was made by using reinforcement learning, for obstacle avoidance, along with an EKF-SLAM algorithm. In [75], an efficient EKF-SLAM algorithm was developed based on measurement clustering. This work tried to address the problem of high computational cost in normal EKF-SLAM at the presence of large number of landmarks.

As for localization, in [73] a cooperative localization approach was proposed through the use of a sequential EKF and convex data. In [160], a map-based localization method for autonomous vehicles was developed using a 3D LiDAR. First, the data from the LiDAR was processed with a curb detection algorithm and a beam model to extract the information. Subsequently, the Iterative Closest Point (ICP) algorithm as well as two Kalman filters were developed to estimate the position of the vehicle based on the output of the LiDAR map.

Many researchers proposed multi-sensor fusion-based localization using EKF. In [82], a multi-sensor fusion scheme based on EKF was proposed for fusing VO produced by two cameras. In [32], an EKF-based multi-sensor fusion scheme was presented for fusing the measurements from a LiDAR sensor with the measurements of a camera and an IMU. The fusion scheme aimed at improving the pose estimation accuracy through multi-sensor fusion. In [98], a generic EKF-based multi-sensor fusion scheme for localization was developed. The algorithm can receive measurements from any arbitrary number of sensors, along with their covariances, and provide the pose estimate.

Unscented Kalman Filter

A more accurate nonlinear extension of the Kalman filter is the Unscented Kalman Filter (UKF), which uses the unscented transform to approximate the probability distributions.

The basic idea of the unscented transform is selecting sample points to represent each distribution. These sample points are then propagated directly through the nonlinear model. After propagation, a new Gaussian distribution is calculated using the propagated samples [158]. UKF deals with nonlinearity and uncertainty propagation in an elegant and more accurate way, without the need for calculating any Jacobians.

One obvious downside of UKF compared to EKF is the increased computational time, due to the sampling and the matrix square root, which are calculated each time-step.

In [84], a robust localization method using adaptive UKF, was introduced. The method used fuzzy inference to adapt the values of its process noise covariance and measurement covariance. In [179], a nonlinear adaptive square root UKF was used to develop a FastSLAM algorithm. The UKF filter replaced the traditional particle filters and Kalman filters; to reduce the particle degradation, linearization errors and eliminate the need of calculating the Jacobians.

In [102], the estimation of the pose of a wheelchair was addressed through fusing the measurements of two encoders, a magnetometer and an accelerometer. The fusion was accomplished using a UKF. In [80], visual measurements were fused with encoders measurements to increase the sampling rate of the visual measurements. In [85], a multi-layer data fusion scheme was proposed. The scheme fused the measurements from encoders, IMU, and LiDAR sensors using three difference stages. First, a UKF is used, followed by an AMCL and a 2D normal distribution transform.

Full Information Estimation

Full information estimation is based on a different approach for formulating the estimation problem. Rather than estimating the states by recursive filters, as in the Bayesian filters, the estimation problem is formulated as a nonlinear optimization problem. The optimization is performed over the whole history of states, controls, and measurements

$$\hat{\mathbf{x}}_{0:T} = \underset{\mathbf{x}_{0:T}}{\arg\max} \quad p(\mathbf{x}_{0:T}|\mathbf{u}_{1:T}, \mathbf{z}_{1:T}), \tag{2.13}$$

where $T \in \mathbb{R}/\{0\}$ is the whole duration of the trajectory being estimated.

The full information estimation is the most general problem in estimation theory. In fact, the previously discussed filters, such as the Kalman filters, are just a special case of this problem when inducing specific assumptions.

In essence, the full information estimation can be categorized into two problems: the Maximum a Posteriori (MAP) estimation and the Maximum Likelihood Estimation (MLE). The difference between the two estimation problems is that MAP takes a prior into consideration while MLE assumes it to be uniform.

Maximum a Posteriori Estimation

For the state vector $\mathbf{x} \in X \subset \mathbb{R}^n$ and the measurement vector $\mathbf{z} \in Z \subset \mathbb{R}^m$, let $\mathbf{X}$ be the whole state sequence

$$\mathbf{X} := (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k) \in X^{k+1}, \tag{2.14}$$

$\mathbf{Z} \in Z \subset \mathbb{R}^m$ be the whole measurements sequence

$$\mathbf{Z} := (\mathbf{z}_1, \ldots, \mathbf{z}_k) \in Z^k, \tag{2.15}$$

and $\mathbf{U} \in U \subset \mathbb{R}^r$ be the whole control sequence through operation. Since the full information optimization problem accepts constraints, the feasible state, control and measurement sets are defined as X, U and Z, respectively. Using these sequences, the MAP problem can be formulated as

$$\mathbf{X}^* := \arg\max_{\mathbf{X}} p(\mathbf{X}|\mathbf{U}, \mathbf{Z}) = \arg\max_{\mathbf{X}} \frac{p(\mathbf{Z}|\mathbf{X}, \mathbf{U})p(\mathbf{X}|\mathbf{U})}{p(\mathbf{Z})}, \tag{2.16}$$

where $p(\mathbf{Z})$ is the measurement total probability, and is just a normalizing factor that does not affect the optimization problem. Hence, Eq. (2.16) can be rewritten as:

$$\mathbf{X}^* := \arg\max_{\mathbf{X}} p(\mathbf{X}|\mathbf{U}, \mathbf{Z}) = \arg\max_{\mathbf{X}} p(\mathbf{Z}|\mathbf{X}, \mathbf{U})p(\mathbf{X}|\mathbf{U}). \tag{2.17}$$

If the prior is uniformly distributed, it has no effect on the optimization problem and can be removed. In this case, the problem becomes an MLE, formulated as

$$\mathbf{X}^* := \arg\max_{\mathbf{X}} p(\mathbf{X}|\mathbf{Z}) = \arg\max_{\mathbf{X}} p(\mathbf{Z}|\mathbf{X}). \tag{2.18}$$

All the estimators discussed so far stem from the MAP formulation, because the prior information is considered. The MAP problem is the basis for almost all optimization-based estimation such as graph-SLAM and bundle adjustment [22].

In [101], the visual SLAM problem is formulated as a bundle adjustment problem. In [39], the SLAM back-end is implemented as a factor graph with the optimization done through the use of Levenberg-Marquardt. In [176], the SLAM problem using a 3D LiDAR data is divided into two parts. First, a LiDAR odometry and then the LiDAR mapping. In both cases, the author formulates the problem as an MLE optimization problem. In [177], the authors introduce a multilayer optimization for the data from an IMU, a monocular camera and a 3D LiDAR.

Least Squares Estimation

Using the MAP formulation, and assuming the Markov assumption as well as white process and measurement noises, Eq. (2.17) can be factorized as

$$\mathbf{X}^* := \arg\max_{\mathbf{X}} p(\mathbf{x}_0) \prod_{i=1}^{k} p(\mathbf{z}_i|\mathbf{x}_i, \mathbf{u}_i)p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i). \tag{2.19}$$

Furthermore, if the process and measurement noises are assumed to be zero-mean additive Gaussian noises, i.e., Eq. 2.1 and Eq. 2.2 becomes

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k, \tag{2.20}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k, \tag{2.21}$$

and the initial, prior and posterior distributions can be written as

$$p(\mathbf{x}_0) = \frac{1}{\sqrt{2\pi}|\mathbf{P}_0|} e^{-\frac{1}{2}(\mathbf{x}_0-\hat{\mathbf{x}}_0)^\top \mathbf{P}_0^{-1}(\mathbf{x}_0-\hat{\mathbf{x}}_0)}, \tag{2.22}$$

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) = \frac{1}{\sqrt{2\pi}|\mathcal{Q}_k|} e^{-\frac{1}{2}(\mathbf{x}_k-\hat{\mathbf{x}}_k)^\top \mathcal{Q}_k^{-1}(\mathbf{x}_k-\hat{\mathbf{x}}_k)}, \tag{2.23}$$

$$p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{u}_k) = \frac{1}{\sqrt{2\pi}|\mathcal{R}_k|} e^{-\frac{1}{2}(\mathbf{z}_k-\hat{\mathbf{z}}_k)^\top \mathcal{R}_k^{-1}(\mathbf{z}_k-\hat{\mathbf{z}}_k)}, \tag{2.24}$$

where $\hat{\mathbf{x}}_0$ is the mean of the initial state distribution, and the means $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{z}}_k$, for zero-mean noises, are defined as

$$\hat{\mathbf{x}}_k := \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k), \tag{2.25}$$

$$\hat{\mathbf{z}}_k := \mathbf{h}(\mathbf{x}_k). \tag{2.26}$$

The maximization problem in Eq. (2.19) is equivalent to the minimization of the log posterior. Therefore, the problem can be written as

$$\mathbf{X}^* = -log(p(\mathbf{x}_0)) + \arg\min_{\mathbf{X}} \sum_{i=0}^{k} -log(p(\mathbf{z}_i|\mathbf{x}_i, \mathbf{u}_i)) - log(p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i)). \tag{2.27}$$

Substituting with the Gaussian distributions in Eq. (2.22), (2.23), and (2.24), then rearranging the equation yields

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \underbrace{\frac{1}{2}||\mathbf{x}_0 - \hat{\mathbf{x}}_0||^2_{\mathbf{P}_0^{-1}} + \frac{1}{2}\sum_{i=1}^{k}\{||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2_{\mathcal{Q}_i^{-1}} + ||\mathbf{z}_i - \mathbf{h}(\mathbf{x}_i)||^2_{\mathcal{R}_i^{-1}}\}}_{J}, \tag{2.28}$$

Eq. (2.28) is the Least Squares Estimation (LSE) cost function. Notice that the only remaining condition, in order for the least squares estimator to become a Kalman filter, is the linearity condition. In fact, if the system is linear, the LSE is equivalent to the Kalman filter.

As can be seen in Eq. (2.28), each of the three terms of the cost function is defined using the subtraction operator. The definition of the error in the cost function using the subtraction operator is only defined for states and measurements over the Euclidean space $\mathbb{R}^d, d \in \mathbb{N}$. However, in case of states which exists in non-Euclidean spaces, such as the rotation manifold, this definition falls apart and a new error function needs to be defined. In this case, the cost function in (2.28) can be written as

$$J(\mathbf{X}, \mathbf{Z}) = \frac{1}{2}||\mathbf{e}_\mathbf{x}(\mathbf{x}_0, \hat{\mathbf{x}}_0)||^2_{\mathbf{P}_0^{-1}} + \frac{1}{2}\sum_{i=1}^{k}\{||\mathbf{e}_\mathbf{x}(\mathbf{x}_i, \hat{\mathbf{x}}_i)||^2_{\mathcal{Q}_i^{-1}} + ||\mathbf{e}_\mathbf{z}(\mathbf{z}_i, \hat{\mathbf{z}}_i)||^2_{\mathcal{R}_i^{-1}}\}, \tag{2.29}$$

where $\mathbf{e}_\mathbf{x} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is an error function over the state estimates, $\mathbf{e}_\mathbf{z} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is an error function over the sensor measurements.

The LSE problem formulation is widely used in SLAM and bundle adjustment problems. A popular example is the graph optimization library g2o [72].

Note that the information matrix (inverse of covariance matrix) is used here as a weighting for each component of the estimation. Eq. (2.29) shows that if $\mathcal{Q} >> \mathcal{R}$, the optimization and in turn the estimation will rely on the measurements from the sensors while neglecting the model prediction. Similarly, if $\mathcal{R} >> \mathcal{Q}$, then the optimization and in turn the estimation will rely more on the prediction model.

The previous formulation for the LSE resulted from the assumption of zero-mean Gaussians. Other assumptions over the distribution will lead to different types of norms in the cost function. For example, assuming Laplacian distributions over the process and measurement noises substitute the l_2 norm with an l_1 norm [22].

2.2.5 Moving Horizon Estimation

Minimizing the cost function stated in (2.29) becomes, in general, intractable for long duration of operation, because of the growth of the number of the optimization variables. Besides, older states effect on current states decreases with time. Accordingly, a Moving Horizon Estimation (MHE) strategy can be adapted as an approximate approach to a full information MAP. To this end, for $N \in \mathbb{N}/\{0\}$, the following finite horizon cost function is defined.

$$J_{MHE}(\mathbf{X}_N, \mathbf{Z}_N) = \frac{1}{2}||\mathbf{e}_\mathbf{x}(\mathbf{x}_{k-N}, \hat{\mathbf{x}}_{k-N})||^2_{\mathbf{P}_{k-N}^{-1}} + \frac{1}{2}\sum_{i=k-N+1}^{k}\{||\mathbf{e}_\mathbf{x}(\mathbf{x}_i, \hat{\mathbf{x}}_i)||^2_{\mathcal{Q}_i^{-1}} + ||\mathbf{e}_\mathbf{z}(\mathbf{z}_i, \hat{\mathbf{z}}_i)||^2_{\mathcal{R}_i^{-1}}\}, \tag{2.30}$$

where N is the estimation horizon,

$$\mathbf{X}_N := (\mathbf{x}_{k-N}, \mathbf{x}_{k-N+1}, \dots, \mathbf{x}_k) \in X^{N+1},$$

$$\mathbf{Z}_N := (\mathbf{z}_{k-N+1}, \dots, \mathbf{z}_k) \in Z^N,$$

are the history of robot states and measurements over the estimation horizon, $||\mathbf{e}_{\mathbf{x}}(\mathbf{x}_{k-N}, \hat{\mathbf{x}}_{k-N})||^2_{\mathbf{P}^{-1}_{k-N}}$ is the arrival cost of the moving horizon problem, and the weighting matrix $\mathbf{P}_{k-N}$ is the estimation covariance of the $(k-N)$-th state, which can be computed using the EKF Riccati equation [121]

$$\mathbf{P}_{k+1} = \mathcal{Q}_k + F_k \mathbf{P}_k F_k^\top - F_k \mathbf{P}_k H_k^\top (\mathcal{R}_k + H_k \mathbf{P}_k H_k^\top)^{-1} H_k \mathbf{P}_k F_k^\top,$$

where $F_k \in \mathbb{R}^{n \times n}$ and $H_k \in \mathbb{R}^{m \times n}$ are the process model and the measurement model Jacobians given by

$$F_k := \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\mathbf{x}_k}, \qquad H_k := \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\mathbf{x}_k}. \tag{2.31}$$

Consequently, the moving horizon optimization problem can be stated as

$$\begin{aligned}
&\min_{\mathbf{X}_N \in X^{N+1}} J_{MHE}(\mathbf{X}_N, \mathbf{Z}_N) && \text{(2.32a)}\\
\text{subject to}\quad &\mathbf{x}_i = \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \;\; \forall\, k-N \le i \le k, && \text{(2.32b)}\\
&\mathbf{x}_i \in X \qquad\quad\; \forall\, k-N \le i \le k, && \text{(2.32c)}\\
&\mathbf{u}_i \in U \qquad\quad\; \forall\, k-N+1 \le i \le k. && \text{(2.32d)}
\end{aligned}$$

Several works proposed the use of MHE in multi-sensor fusion localization. In [94, 132], a nonlinear MHE was used for relative localization of a multi robot system. An efficient algorithm based on real-time iteration scheme was proposed, to solve the high computation problem. In [141], a distributed MHE for a mobile robot localization problem was introduced, using sensor networks. The estimator stability and the approximation of the arrival costs were discussed.

In [66], a localization algorithm was developed, which fuses the measurements of a laser range finder along with the wheel odometry data of an autonomous vehicle. The optimization problem was formulated as an MHE problem (as in Eq. 2.32) and solved using Sequential Quadratic Programming (SQP).

In [83], a multi-rate MHE was developed for localization through fusing the data from sensors having different sampling times. The proposed MHE used an inertial sensor and a camera with different sampling rates. The constrained optimization problem was solved using interior point algorithm. Furthermore, the input-to-state stability of the estimator was studied in the presence of bounded disturbances and noises.

In [66], the authors combined an EKF with MHE for three-dimensional underwater localization. The algorithm achieved a compromise between better accuracy and lower computational requirement. MHE was used to fuse a laser range sensor measurement and the odometry information of a vehicle to achieve more robust localization against outliers in the laser range sensor measurements.

Similarly, in [37], a multi-rate MHE sensor fusion algorithm in the presence of time-delayed measurements and missing measurements was developed. A computationally efficient implementation of linear MHE was proposed, only for the case where an analytical solution of the linear MHE problem was be found.

In most of the aforementioned work, the proposed algorithms were based on linear measurement and system models to avoid the high computational cost of the MHE implementation. Furthermore, the solutions proposed were case specific and, in most cases, cannot be generalized to arbitrary number of sensor measurements.

2.3 Control of Autonomous Mobile Manipulators

The separate control of mobile robots and robotic arms was studied extensively in the literature. The considered control problems can be categorized under point-stabilization, trajectory tracking, and path following. The commonly used control techniques include feedback linearization [30], robust control [69], fuzzy-based feedback linearization [115], adaptive control [117, 142], and Model Predictive Control (MPC) [43, 95, 167].

Several studies that consider controlling mobile manipulators as a combined system also exist. For example, in [139], the authors designed a whole-body controller based on feedback linearization, which controls the end-effector pose of a mobile manipulator; the controller was tested on a 7-DOF mobile manipulator. In [113], an adaptive back-stepping control for the trajectory tracking of mobile manipulators was proposed. In [97], a robust nonlinear controller with uncertainty estimator was developed; the controller was validated through simulations for a 7-DOF mobile manipulator. Furthermore, in [11], linear MPC for developing a reactive constrained controller of an omni-directional mobile base with a 5-DOF robotic arm was used. All the aforementioned studies considered mobile manipulators with non-redundant arms. This simplifies the problem, due to the presence of a closed-form inverse kinematics solutions for such arms. Thus, the control of the mobile manipulator end-effector can be designed in the configuration (joint) space by utilizing a separate closed-form inverse kinematics module, see, e.g. [11].

MPC is popular in the field of controls because of its ability to handle constrained possibly nonlinear multi-input-multi-output (MIMO) systems. In MPC, a cost function characterizing the control objective is minimized using an open-loop control sequence or function while state and control constraints are considered. The first part of the resulting open-loop control is then applied to the system. Finally, the process is repeated every decision instant, see, e.g. [8].

Chapter 3

Visual Odometry Enhancement[1]

3.1 Introduction

The drawback of Visual Odometry (VO) is the reliance on integration, which means that if during operation, the incremental motion estimation diverged, the overall pose estimation will diverge. This could happen due to, for example, false matched features, bad lighting and illumination problems, random noise or motion bias [170].

To overcome the above mentioned issues, in this chapter, a generic and modular image processing pipeline is proposed to enhance the accuracy and robustness of feature-based VO algorithms, which can be applied to any type of VO; monocular, RGB-D or stereo vision.

The proposed pipeline includes additional filtration and pre-processing stages through CLAHE filter with adaptive thresholding to overcome lighting changes [124]. Additionally, the SSC is used to avoid any bias in the motion estimation [12]. Finally, a novel outlier rejection algorithm referred to as Angle based Outliers Rejection (AOR) is proposed, to reject false matched features as well as features captured on a moving object in the scene. The pipeline is integrated into a stereo, RGB-D, and monocular VO and validated using KITTI [49] and TUM datasets [144], as well as experimental sequences generated by an omni-directional mobile robot.

The contribution of this chapter is integrating the CLAHE filter and the SSC algorithm as well as the proposed AOR algorithm to the VO. The results show that the three stages

[1]This chapter is under submission at the *IEEE Trans. on Intelligent Transportation Systems* as Mohamed Sabry, Mostafa Osman, Ahmed Hussein, Mohamed W. Mehrez, Soo Jeon, and William Melek, "A Generic Image Processing Pipeline for Enhancing Accuracy and Robustness of Visual Odometry.".

play an integral part to enhancing the performance of VO while overcoming the drawbacks of every individual stage.

The remainder of this chapter is organized as follows, Section 3.2 explains the proposed pipeline with the different filtration steps. The experimental work is presented in Section 3.3 which also includes the implementation details, along with the used datasets and the evaluation metrics. Section 3.4 presents the results and discussions. Finally, Section 3.5 provides concluding remarks.

3.2 Proposed Pipeline

In this section, the components of the proposed image processing pipeline are introduced. The flow chart of the pipeline is shown in Fig. 3.1.

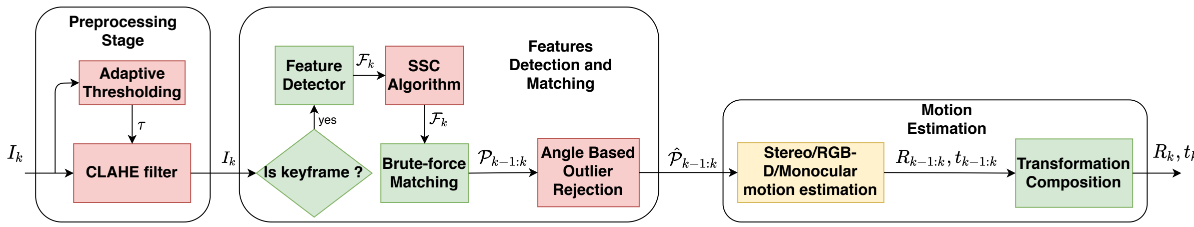


Figure 3.1: Proposed pipeline for the robust feature-based VO with the added filtration stages highlighted in red

3.2.1 Image Pre-processing

The pre-processing stage consists of applying a simple blurring filter to remove some of the noise in the image followed by applying an Adaptive Histogram Equalization (AHE) technique, namely CLAHE [184]. CLAHE is applied to each input frame to enable the feature detector to find sufficient number of features per frame. Although traditional AHE techniques tend to over-amplify the noise in the nearly constant regions in an image, CLAHE filter prevents this over-amplification by limiting the histogram values. The effect of the CLAHE is shown in Fig. 3.2.

Moreover, to ensure that the CLAHE filter adapts to different lighting conditions, the threshold value of the CLAHE is made adaptive to the ratio between the minimum, maximum, and the median of the intensity values in the frame, as presented in (3.1). The

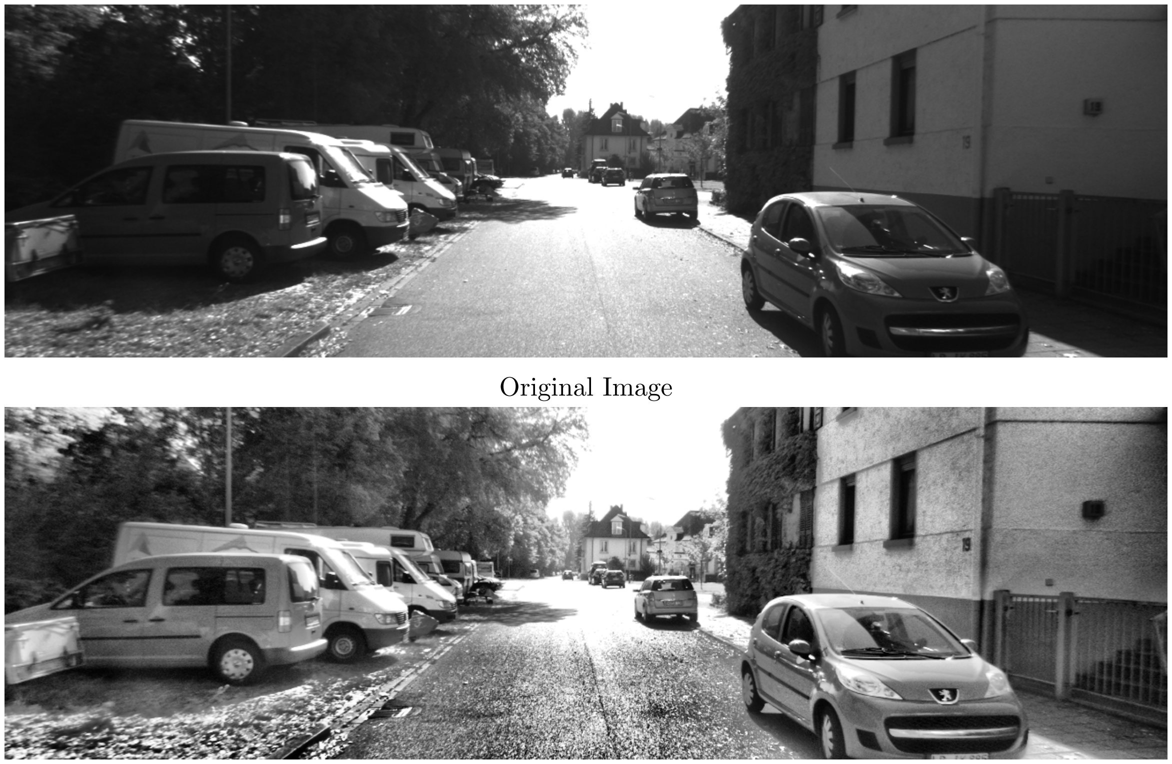


Figure 3.2: The effect of CLAHE filter on the image. Top: The image before applying CLAHE, Bottom: The image after applying the CLAHE algorithm with the adaptive thresholding [The image was taken from KITTI dataset]

adaptation of the threshold value enables the CLAHE filter to adapt to different lighting conditions during the mobile platform operation and to avoid deterioration of the VO performance, caused by too high or too low brightness in the images. Specifically, the contrast at which the CLAHE filter clips the histogram, is computed as

$$\tau_k = \frac{\max(I_k) - \min(I_k)}{\text{median}(I_k)} \tag{3.1}$$

where τ_k and I_k are the contrast threshold for the CLAHE filter, and the 2D image data at the k-th time step, respectively.

An example of the output of CLAHE filter is shown in Fig. 3.2. The effect of the sun can be seen in the original image which leads to bright regions in the top middle of the image and dark regions on the left and the right of the image. Applying CLAHE results in decreasing the effect of the sun on the image and increasing the amount of information, which can then be used by the feature detector algorithm to capture more stable features from the image.

3.2.2 Features Detection and Matching

After the image pre-processing, the current image I_k is passed to a feature detector. The extracted set of features, denoted by $\mathcal{F}_k$, is then matched with the set of those from the previous frame $\mathcal{F}_{k-1}$. Then, the set of matched features $\mathcal{P}_{k-1:k}$ is used for estimating the incremental motion between the two images I_{k-1} and I_k.

One of the main causes of error in the motion estimation is the poor distribution of features associated with the image [170, 5]. Another cause of poor motion estimation is the presence of outliers in the detected and matched features. Therefore, in this proposed work, we added the SSC as well as the proposed AOR steps to the pipeline.

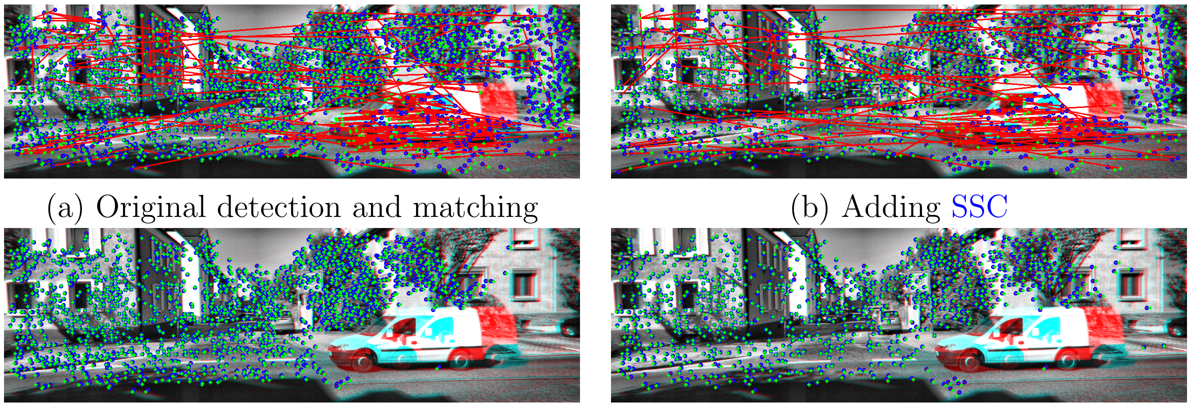
Suppression via Square Covering

Before passing the feature set $\mathcal{F}_k$ to the feature matching algorithm, the features are first passed to the SSC [12] algorithm to make sure the captured features are homogeneously distributed over the whole captured image I_k.

The SSC algorithm is an approximation of the Suppression via Disk Covering (SDC). It relies on an approximate nearest neighbor algorithm, which uses a randomized search tree. In contrast, the SSC achieves comparable results with a single query operation per search range guess. Accordingly, the SSC has a better efficiency and scalability over the SDC. Besides, SSC applies square approximation for the SDC to avoid computing the Euclidean distance between a large number of features. This allows the SSC algorithm to execute in runtime with lower complexity as the number of features increase.

The effect of using the SSC algorithm is shown in Fig. 3.3. Fig. 3.3-(a), shows the original output of SURF feature detector where the feature density is higher in the top right region of the image. As shown in Fig. 3.3-(b), after applying the SSC, the density of the features is almost the same across the image (except for regions which did not contain any features).

Although several feature matching algorithms were introduced in the literature [100], those algorithms generate a considerable amount of false matched points (as can be seen


(a) Original detection and matching (b) Adding SSC

(c) Adding AOR (d) Adding SSC and AOR

Figure 3.3: The effect of SSC, and AOR on the features detection and matching. The previous frame is shown as the red component of I_{k-1} and the current frame is shown as the blue component of I_k. The green crosses (+) and the blue circles (○) represent the features $\mathcal{F}_{k-1}$ and $\mathcal{F}_k$ respectively. Finally, the red lines represent the matched pairs in $\mathcal{P}_{k-1:k}$. Each of the images represents the following: (a) the original features pairs through using SURF detector and a brute-force matching algorithm, (b) shows the features pairs after adding the SSC algorithm only. (c) shows the features pairs after adding the AOR algorithm only, and finally (d) shows the features pairs after adding both SSC and AOR.

in Fig. 3.3-(a) and Fig. 3.3-(b)). Motivated by this issue, in this chapter, a novel outlier rejection algorithm is introduced and integrated to the overall VO pipeline to remove those false matched points.

Angle based Outliers Rejection (AOR) for feature matching

To filter the produced matched points from outliers, a new outlier rejection algorithm called the AOR is proposed. In addition to removing false matched features, the AOR can remove features detected on dynamic objects in the scene, as long as the dynamic object motion is different from that of the vehicle.

Usually, during the motion of the camera, the farther the feature from the center of the image (the principal point), the more the feature moves. Although the amount of movement of a feature in the successive images is different depending on its position, it should be comparable to the motion of other features. False matched points tend to show a larger amount of feature motion through the image as shown in Fig. 3.3-(a). The AOR uses the distance traveled by the feature in the successive images along with the actual position of the feature in those images to remove false matched points.

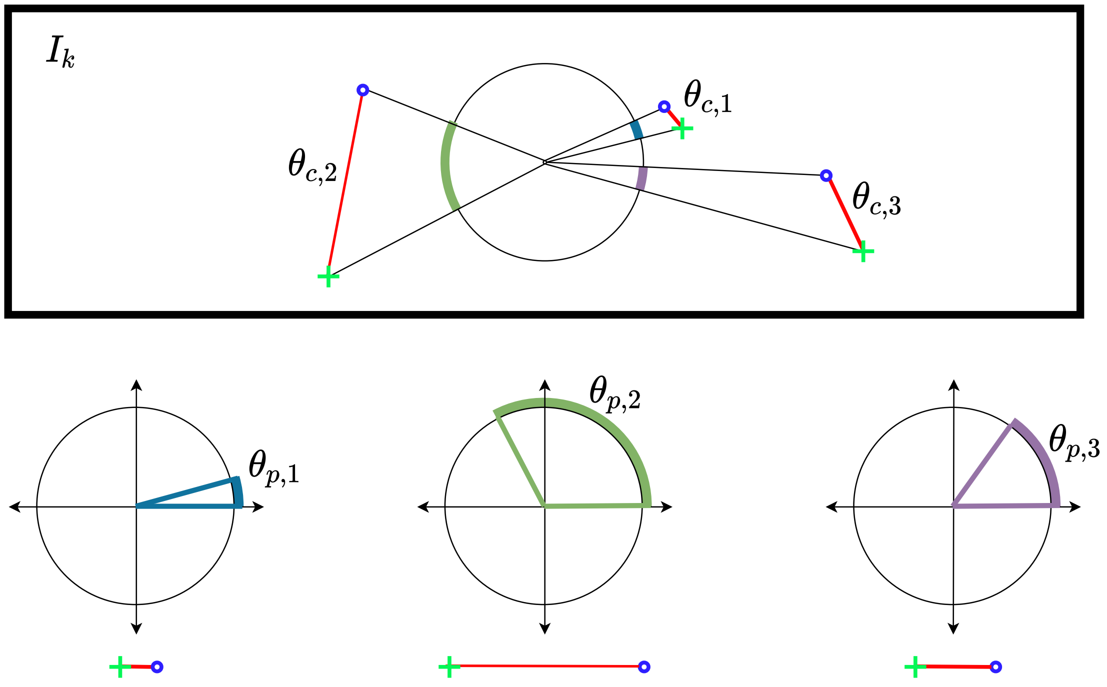


Figure 3.4: Visual illustration of θ_c and θ_p for three different features pairs in the an image.

Fig. 3.4 illustrates the two metrics used by the proposed AOR. AOR can be divided into two steps. First, the angle θ_c between the lines drawn from the center of the image to the feature (shown in Fig. 3.4) in I_{k-1} and I_k is simply calculated as [164]

$$\theta_{c,i} = \arccos \frac{x_{k-1,i} \cdot x_{k,i} + y_{k-1,i} \cdot y_{k,i}}{\sqrt{x_{k-1,i}^2 + y_{k-1,i}^2} \cdot \sqrt{x_{k,i}^2 + y_{k,i}^2}}, \tag{3.2}$$

where $(x_{k-1,i}, y_{k-1,i})$ and $(x_{k,i}, y_{k,i})$ are the coordinates of the i-th feature in the frames I_{k-1} and I_k, with respect to the center of the image, respectively. Notice that θ_c represents the amount of feature motion irrespective of its position in the frame. θ_c for different feature positions is illustrated in the top plot of Fig. 3.4.

Second, the Euclidean distance traveled by the feature projected on a reference circle as shown in the bottom plot of Fig. 3.4, and the corresponding angle θ_p are calculated as [163]:

$$E_i := \left\| \begin{bmatrix} x_{k,i} \\ y_{k,i} \end{bmatrix} - \begin{bmatrix} x_{k-1,i} \\ y_{k-1,i} \end{bmatrix} \right\|_2, \tag{3.3}$$

$$\theta_{p,i} = \frac{E_i}{R} \tag{3.4}$$

where R is the radius of the reference circle. Notice that R determines the sensitivity of the values of θ_p. The larger the radius, the smaller the angle for larger Euclidean distances. The radius can be calculated as,

$$R = \sqrt{\frac{c_x^2 + c_y^2}{\zeta}}, \tag{3.5}$$

where c_x and c_y are the centers of the image, and ζ is a parameter which controls the size of the radius. During the experimentation, the best results were obtained by $\zeta = 8$, however, different values may work better for different conditions.

Using the two angles θ_c and θ_p, a score S is computed for each feature as:

$$S_i = |\theta_{c,i}\theta_{p,i}(\theta_{c,i} - \theta_{p,i})|. \tag{3.6}$$

Notice that for every matched feature, the greater θ_p and θ_c values and the difference between them, the larger the score AOR yields.

Finally, a feature is selected as an inlier, if its AOR score value is less than a threshold η calculated as

$$\eta = c \cdot \text{median}\,(\mathcal{S}), \tag{3.7}$$

where $\mathcal{S}$ is the score set of the matched features, and $c > 1$ is a parameter, which is set to 2 in this work (through tuning). The overall AOR algorithm is summarized in Algorithm 3.

Fig. 3.3-(c) shows the effect of AOR on the detected features. By using AOR, all the outliers, which are present in Fig. 3.3-(a), are removed, and only the true features describing the motion of the camera remain. Furthermore, notice the effect of the AOR algorithm in removing the features detected on the moving vehicle present in the image, since the motion of such features does not agree with the motion of the remaining features. Fig. 3.3-(d) shows the effect of both SSC and AOR on the image, where the inliers remaining in the image are better distributed due to the SSC effect.

Algorithm 3: AOR Algorithm

```
Set R and $\eta$
for $p_i \in \mathcal{P}_{k-1:k}$ do
    Calculate $\theta_{c,i}$ as in Eq. (3.2).
    Calculate the euclidean distance of the feature motion $E$ defined in (3.3).
    Calculate $\theta_{p,i}$ as in Eq. (3.4).
    Calculate the feature AOR score $S$ as in Eq. (3.6).
    Push $S_i \rightarrow \mathcal{S}$.
end
Calculate $\eta$ as in Eq. 3.7.
for $p_i \in \mathcal{P}_{k-1:k}$ do
    if $S_i < \eta$ then
        $p_i \rightarrow \hat{\mathcal{P}}_{k-1:k}$
    end
end
where $\hat{\mathcal{P}}_{k-1:k}$ is the set of matched features inliers.
```

The filtered matched feature set $\hat{\mathcal{P}}_{k-1:k}$ can then be passed to any VO algorithm to estimate the incremental motion of the camera and to compute the odometry.

3.3 Experimental Works

To show the generic aspect of the proposed pipeline, a simple stereo, RGB-D, and monocular VO algorithms are implemented for validation. The motion estimation techniques used are the same as those described in [133].

The algorithms are implemented in Python using OpenCV library, SURF for feature tracking, and the extracted features are matched between consecutive frames by brute-force search. All experiments and tests were conducted on a computer with an Intel i7-8850H 6-core processor running at 2.60 GHz using 16 GB of RAM, running Ubuntu 16.04. Furthermore, the algorithms were implemented with a ROS wrapper node, to be compatible with ROS framework [118].

The VO algorithms are then used to estimate the motion of the camera using sequences from KITTI [49], TUM [144] as well as experimental sequences generated by Summit-XL Steel manufactured by Robotnik Inc. [126]. The performance of the pipeline is evaluated through several comparisons which demonstrate the effect of the added stages to the VO pipeline.

3.3.1 Motion Estimation

Stereo / RGB-D Visual Odometry

The stereo and RGB-D VO algorithms used in this chapter rely on solving the same 3D-to-2D correspondence problem. First, the features in the frame I_{k-1} along with the disparity map or the depth image are used to produce the 3D features F_{k-1} in I_{k-1}. The motion of the camera is then estimated by solving the Perspective-n-Point (PnP) problem. The PnP problem is solved in a RANSAC scheme to achieve better motion estimation [47].

$$T_{k-1:k} = \underset{T_{k-1:k}}{\arg\min} \sum_{i=0}^{N_f} \left\| f_k^i - \hat{F}_{k-1}^i \right\|_2^2 \tag{3.8}$$

$T_{k-1:k} \in SE(3)$ is the transformation matrix describing the incremental motion between time-steps $k-1$ and k, f_k^i is the i-th 2D feature in the current frame, $\hat{F}_{k-1}^i$ is the same feature in 3D, reprojected from frame I_{k-1} onto the current frame I_k through $T_{k-1:k}$, and N_f is the total number of features in the frame.

Monocular Visual Odometry

To estimate the motion from a single camera, the epipolar constraint between frames is used, as

$$\begin{bmatrix} x_{k-1,i} & y_{k-1,i} \end{bmatrix} \mathbf{E} \begin{bmatrix} x_{k,i} \\ y_{k,i} \end{bmatrix} = 0, \quad \forall\ 0 \le i \le N_f \tag{3.9}$$

where $\mathbf{E} \in \mathbb{R}^{3\times 3}$ is the essential matrix for the calibrated camera [53].

The essential matrix $\mathbf{E}$ is estimated using the five-point algorithm proposed in [104]. After obtaining the essential matrix, it is decomposed to the translation and rotation of the camera as described in [53]. Furthermore, the motion estimation algorithm is executed in the least median of squares (LMEDS) [128] scheme in order to achieve better motion estimation.

Using a monocular camera, the ego-motion of the camera can be estimated up to a scale. To compensate for this scale, a velocity measurement of the vehicle needs to be available through the use of an external sensor such as wheel encoders, an IMU, a GPS, or through the CAN data from the vehicle's tachometer.

The implementation of the VO algorithms from scratch was intended for the ease of integration of the proposed pipeline. However, the pipeline, in general, can be integrated to any VO implementation.

3.3.2 Datasets

KITTI Vision Benchmark Dataset

KITTI Vision Benchmark Suite was selected as a publicly available dataset [49]. The dataset provides ground-truth ego-motion for 11 training sequences and 11 test sequences. The ground-truth is provided as a list of 6D poses for the training sequences, whereas for the test sequences evaluation results are obtained by submitting to the KITTI website. The dataset is sampled at 10 Hz at an average speed of 90 km/h, which creates a challenge in using the dataset for training and testing. Sequence 3 from the training subset is no longer available, as it was removed by KITTI for its similarities with the test sequences.

The dataset comprises the following information: raw synced and rectified color images from the left and right cameras and raw 3D GPS/IMU unfiltered data, along with the timestamps for all recordings. In order to convert the raw data to ROS bagfiles, *kitti2bag* package was used [70]. The dataset also provides a tool for evaluating the performance of the VO and visual SLAM algorithms. This tool was used in the chapter for evaluating the proposed pipeline in the case of KITTI dataset.

TUM RGB-D Dataset

TUM RGB-D dataset is a large dataset containing sequences captured by an RGB-D camera along with its ground-truth to establish a benchmark for evaluation of VO and visual SLAM algorithms [144]. The dataset contains the color and depth images taken by a Microsoft Kinect camera while the ground-truth was recorded using a high-accuracy motion capture system with eight high-speed tracking cameras (100 Hz). The data was recorded using a 30 Hz rate with the camera resolution (640×460). The dataset also provides an online tool through which the results are submitted for evaluating the performance of VO and visual

SLAM systems. In this chapter, the TUM sequences are evaluated using the Relative Pose Error (RPE) which is recommended by the dataset for VO algorithms [151].

RPE is basically the error in relative motion between the pairs of the VO output. The evaluation tool by TUM dataset computes the error between all pairs of the output and generates the evaluation metrics such as the Root Mean Square Error (RMSE), mean, max, etc. In this chapter, the RMSE error for the translation and orientation is used for evaluation.

Images from Omnidirectional Robot

Summit XL Steel is a ground mobile robot with meccanum wheels shown in Fig. 3.5. The robot is equipped with a Velodyne LiDAR sensor [154], an Astra RGB-D Camera [10], a Pix-hawk 3 auto-pilot used as an IMU [116], and wheel encoders. Several experiments were made using the robot to validate the proposed pipeline, while using the VICON motion capture system as a reference [155]. The VICON system used, consists of 12 cameras and the VICON bridge package was used to couple VICON with ROS [2]. Since the RGB-D VO case is tested and validated using TUM dataset, the Summit-XL Steel sequences are used to validate the monocular VO, while relying on the wheel encoders for getting the speed for motion scaling. The evaluation is again conducted using the RMSE error for translation and orientation.

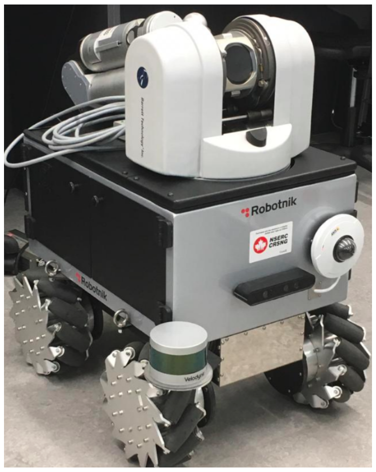


Figure 3.5: The omni-directional mobile robot used for validating the monocular VO algorithm with the proposed pipeline.

Three different sequences were executed using the robot in remote-control mode. In the first two sequences, the robot moved in a semi-rectangular paths while, in the third sequence, the robot moved in a circular path. The total length of each of the paths were 12.5 meters in case of the rectangular paths and 6.5 meters in case of the circular path.

3.4 Results and Discussion

3.4.1 KITTI Vision Benchmark Dataset / Stereo VO

Pose Accuracy Comparison

In order to show the efficacy of the proposed algorithm, the pose estimation results from a stereo VO are reported with and without the proposed pipeline. Table 3.1 shows the accuracy comparison using the 10 sequences available from the KITTI dataset. The results shown are the translation and rotation RMSE values generated by the dataset evaluation tool. As can be seen in the table, the pipeline enhanced the pose estimation accuracy in almost all the sequences.

Table 3.1: KITTI Dataset Accuracy Comparison

Sequence Number	**VO without Pipeline**		**VO with AOR**		**VO with Pipeline**	
	Tr (%)	**Rot** (\deg/m)	**Tr** (%)	**Rot** (deg/m)	**Tr** (%)	**Rot** (deg /m)
0	1.85	0.011	1.80	0.011	**1.68**	**0.011**
1	5.14	0.013	5.33	0.013	**4.72**	**0.012**
2	6.99	0.036	1.82	0.016	**1.89**	**0.017**
4	3.91	0.007	2.75	0.006	**0.33**	**0.007**
5	**2.38**	0.011	2.75	0.012	2.43	**0.009**
6	**2.62**	0.010	2.74	0.010	2.88	**0.010**
7	2.11	0.016	1.95	0.014	**2.07**	**0.015**
8	2.57	0.012	2.65	0.011	**1.92**	**0.012**
9	2.70	0.019	2.86	0.019	**1.78**	**0.019**
10	2.65	0.020	2.82	0.02	**2.02**	**0.020**
Overall	3.29	0.015	2.71	0.013	**2.18**	**0.013**

In sequence 2 (shown in Fig. 3.6), note that the effect of the pipeline is very obvious since the presence of the pipeline significantly enhanced the pose estimation accuracy compared

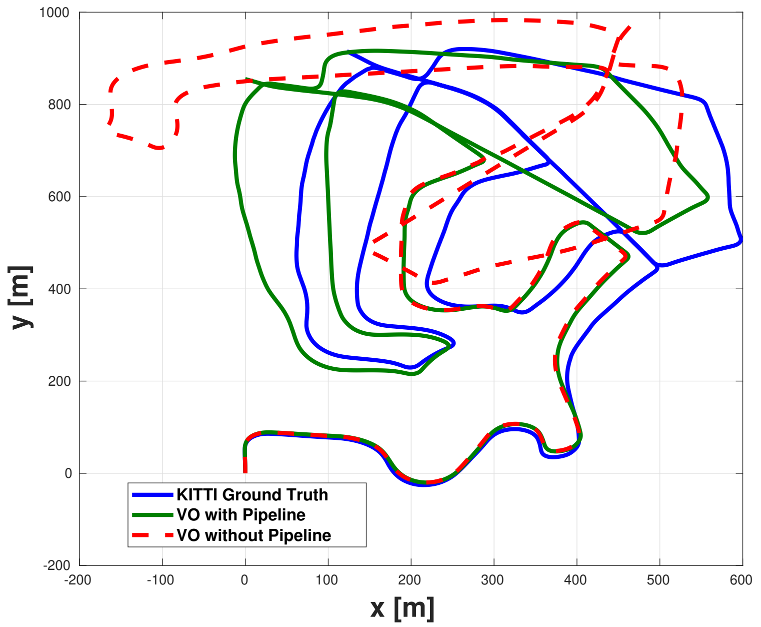


Figure 3.6: The pose estimation results of sequence 2 in KITTI dataset with and without pipeline along with the dataset ground-truth. The proposed pipeline significantly enhances the output accuracy of the VO is this sequence. The VO output without the pipeline suffers from major drift errors and can even be considered to diverge.

to the VO pose estimation without the pipeline. Notice that the reason for the divergence of the VO in the first case is due to the absence of enough features in the images, which made the VO unable to estimate the incremental motion for long durations in the sequence. On the other hand, using the CLAHE filter increases the number of stable features in the images, while using the AOR algorithm along with the RANSAC (which is present in both cases), ensures more accurate incremental motion estimation for all received images. This leads to a much better VO output as shown in Fig. 3.6.

In sequence 5, although the average translation RMSE of the VO without the pipeline is lower than that with the pipeline, the actual performance of the pose estimation for the VO with the pipeline is much better (as shown in Fig. 3.7). The real performance of the VO odometry is not reflected in Table 3.1 because the drift in the orientation of the VO without the pipeline causes some estimated poses to look closer to the ground truth compared to the VO output with the pipeline. However, the overall path estimated by the

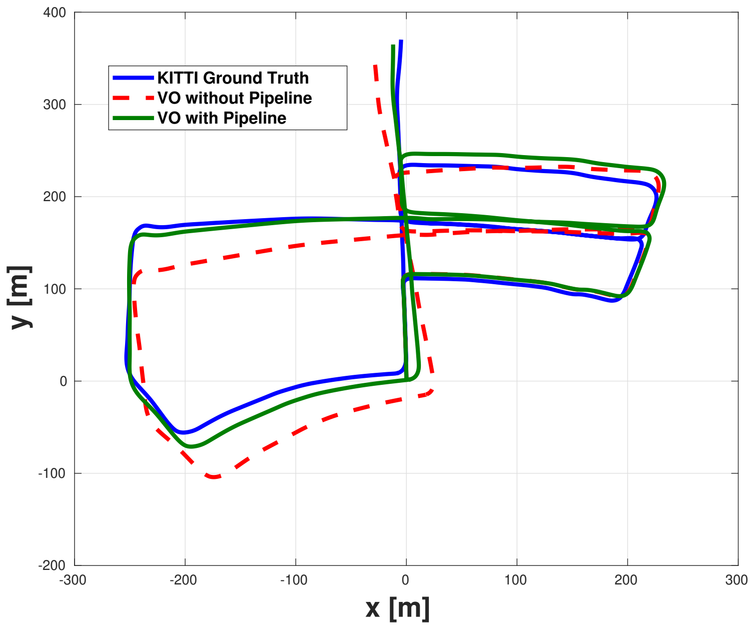


Figure 3.7: The pose estimation results of sequence 5 in KITTI dataset with and without pipeline along with the dataset ground-truth. The performance of the VO with the pipeline enhances the performance of the VO in both translation and orientation estimation. This is not reflected in the average RMSE of the sequenec due to the drift in the orientation of the VO without the pipeline which causes some estimated pose to look closed to the ground truth although the overall estimated trajectory is much worse.

VO with the pipeline is superior to that of the VO without the pipeline.

Finally, in the case of sequence 6, the performance of the VO without pipeline outperformed that of the VO with pipeline. This may be attributed to the fact that the amount of features available after applying the AOR is not enough for accurate motion estimation. This is further discussed in Section 3.4.1.

Effect of AOR

One of the contributions of the chapter is the new outlier rejection algorithm named AOR. In this subsection, the effect of AOR alone on the performance of the VO is studied. To this end, the translation and orientation RMSE results for the VO with AOR are also reported.

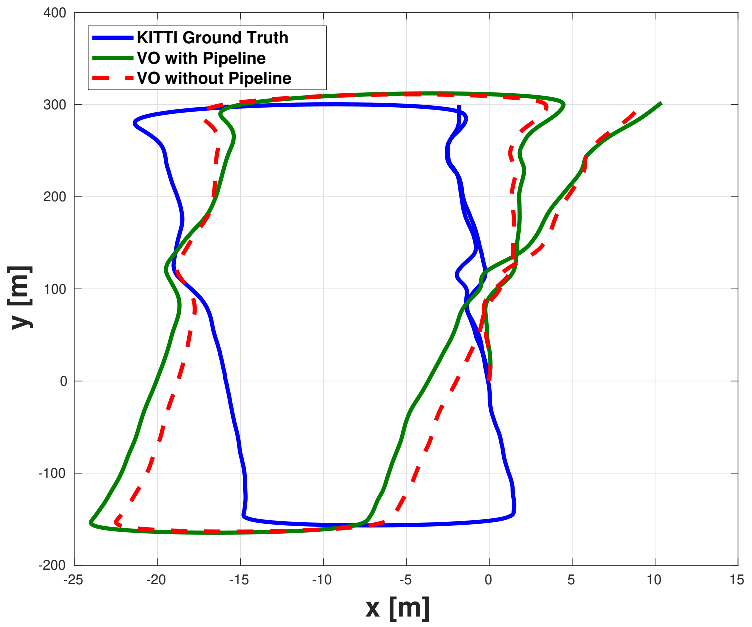


Figure 3.8: The pose estimation results of sequence 6 in KITTI dataset with and without pipeline along with the dataset ground-truth. The VO without pipeline shows more accurate pose estimation compared to VO with pipeline. However, the performance of both algorithms are very similar.

As shown in Table 3.1, the AOR significantly contributed to the enhancement of some of the sequences. For example, the table shows that the use of AOR was responsible for the enhancement of sequence 2 which diverged without the use of it as shown in Fig. 3.6. Furthermore, the use of AOR also resulted in better results for sequences 0, 4, and 7.

As can be seen in the results, the AOR algorithm is an aggressive outlier rejection method. In other words, the AOR might result in the removal of matched features with slight deviations. This means that the use of AOR on a frame, requires the presence of a sufficient amount of stable features for motion estimation. Otherwise, the AOR will lead to the deterioration of the motion estimation due to decreasing the amount of matched features. This is why the integration of AOR with CLAHE and SSC is a very good combination because each of the three stages plays a role in enhancing the motion estimation while complementing the negative effects of the other ones.

Although the use of AOR can lead to the removal of many matched features, the presence of CLAHE increases the amount of stable features in the frames (an example is shown in Fig. 3.10). Moreover, despite the increase in features due to CLAHE, this might lead to a concentration of features in a certain region of the image. However, the use of SSC prevents such poor distribution of the features. Although the use of CLAHE, while adding more stable features, can also add more outliers, the AOR acts on removing these outliers. In conclusion, the presence of the three stages of the pipeline acts as a desirable combination to overcome the drawbacks of each of the stages while utilizing their advantages.

It is worth mentioning that the AOR resulted in worse results for some of the sequences, yet, the overall performance of the VO with the pipeline (2.71%) was better than that of the VO without the pipeline (3.29%). Furthermore, the use of the complete pipeline still resulted in better accuracy for almost all sequences compared to the VO without pipeline as shown in Table 3.1.

Computational Cost

It is expected that adding processing stages to the VO algorithm will lead to an increase in computational time. This can indeed be seen in Table 3.2, where the average computation time for the VO is reported after adding each of the proposed stages of the pipeline. As expected, the computational time of the VO algorithm increases with every step added. However, the significant benefits in the accuracy and the pose estimation can easily outweigh the increased computational cost.

In Table 3.2, the computation time of the VO algorithm with the AOR algorithm only is reported. Notice that the computation time of the algorithm after adding the AOR to the VO is less than that of the VO without AOR. As discussed in Section 3.2, this is due to the fact that the AOR removes the false matched features. Accordingly, this simplifies the mission of the RANSAC which results in a faster motion estimation convergence, and a faster performance as shown in Table 3.2. This also explains why the full pipeline computational time is less than the case of adding CLAHE only or CLAHE and SSC.

The mentioned average computational time shows that the algorithm is capable of working in real-time while receiving up to 6 fps. The performance of the algorithm, as well as the accuracy of the pose estimation, can be further enhanced through the use of a Graphical Processing Unit (GPU) and mutli-threading processing.

Table 3.2: Computation Time Comparison For KITTI Sequences.

VO Type	**Comp. Time** (mean ± std [ms])	**Tr RMSE** (%)	**Rot RMSE** (deg /m)
Vanilla	116 ± 31	3.29	0.0154
CLAHE	176 ± 36	2.88	0.0136
CLAHE and SSC	181 ± 34	2.86	0.0136
AOR Only	**106 ± 24**	2.71	0.0134
Full Pipeline	160 ± 35	**2.18**	**0.0133**

3.4.2 TUM RGB-D Dataset / RGB-D VO

Pose Accuracy Comparison

Table 3.3 shows the translation and orientation RMSE for 9 sequences from TUM RGB-D dataset. As shown in the Table, the RGB-D VO with the proposed pipeline shows better pose estimation accuracy for all sequences. This enhancement varies from one sequence to another based on the lighting, and the number of features available in each sequence.

Table 3.3: TUM Accuracy Comparison

Seq. Name	**VO without Pipeline**		**VO with Pipeline**	
	Tr (m)	**Rot** (deg)	**Tr** (m)	**Rot** (deg)
fr1/xyz	0.24	8.80	**0.04**	**2.08**
fr1/desk	0.43	19.5	**0.07**	**3.55**
fr1/desk2	0.46	23.8	**0.09**	**6.74**
fr1/room	0.32	23.5	**0.12**	**5.73**
fr2/pioneer_360	0.18	6.50	**0.10**	**3.58**
fr2/pioneer_slam	0.10	3.60	**0.09**	**2.38**
fr2/pioneer_slam2	0.07	2.82	**0.07**	**2.00**
fr2/pioneer_slam3	0.07	2.03	**0.05**	**1.44**
fr2/desk	0.19	5.20	**0.02**	**0.64**
Overall	0.23	10.6	**0.07**	**3.12**

Fig. 3.9 shows an example of the pose estimation output from the VO algorithm with and without the pipeline. It can be seen that the VO without the proposed pipeline suffers from large motion estimation errors at the beginning of the path, which in turn results in large drift errors over the rest of the path. As for the VO with the pipeline, although

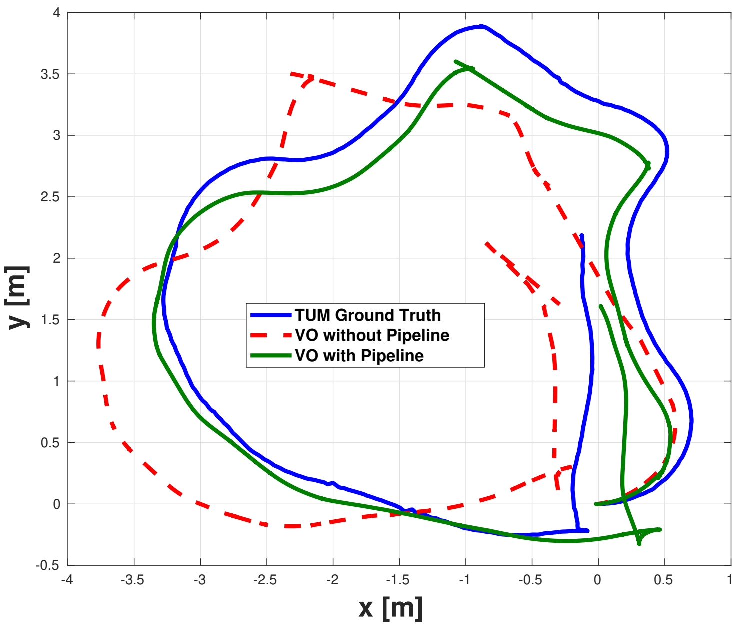


Figure 3.9: The pose estimation results for the sequence fr1/pioneer_360 in TUM dataset with and without pipeline along with the dataset ground-truth. The pose estimation accuracy of the VO with pipeline is superior while the VO without the pipeline suffers from both poor translation and orientation estimation.

there is still an error in the estimated path, the error is significantly smaller than that of the VO without the pipeline especially at the beginning of the path causing a much better estimation for the rest of the path. The better performance of the VO with the pipeline algorithm is attributed to the increased amount of detected features at the beginning of the path compared to that of the VO without the pipeline.

Since the TUM sequences are recorded indoors, adding the CLAHE stage results in a significant increase in the detected features (some examples are shown in Fig. 3.10). Note that for these sequences an RGB-D VO algorithm is used, which means that only the features with an observable depth by the depth sensor can be used for motion estimation. In other words, the far features in the images cannot be used. Through the SSC algorithm, the features detected in the images are well distributed and thus the number of close features with observable depth increases (see Fig. 3.10).

The results shown in Fig. 3.9 and Table 3.3 confirm the efficacy of the proposed pipeline

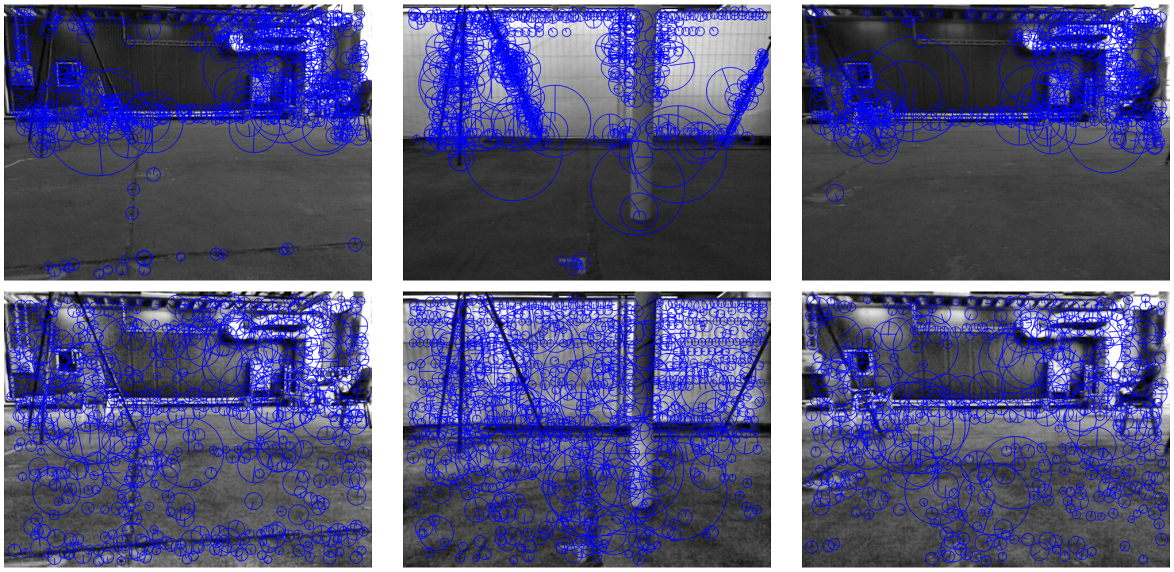


Figure 3.10: Three examples from TUM sequences showing the effect of CLAHE and SSC in indoor scenarios. The images in the top row shows the images with the detected features using SURF detected without the use of CLAHE and SSC algorithm. The images in the bottom row shows the features detected by SURF after adding the CLAHE and SSC stages.

for VO algorithms even for indoor scenarios.

Computational Cost

Table 3.4 shows the average computational time for the VO with the different added stages. Notice that in this case, the results are slightly different from those of the computation cost analysis shown for KITTI sequences. As expected, the computational cost increases for the different stages of the proposed pipeline. However, in KITTI sequences, adding the AOR algorithm to the pipeline resulted in a faster performance compared to the VO without any stages. This is not the case for TUM sequences, where the computational cost still increases with the AOR algorithm. It is postulated that the amount of features detected in the case of TUM is larger or the number of outliers in the matched features set is larger. Nevertheless, adding the AOR to the pipeline, results in lowering the computation cost of the VO algorithm. As can be seen in Table 3.4, the average computational cost of the VO algorithm when adding CLAHE and SSC is larger than that of the overall pipeline computational cost.

Table 3.4: Computation Time Comparison For TUM Sequences.

VO Type	Comp. Time (mean ± std [ms])	Tr RMSE (m)	Rot RMSE (deg)
Vanilla	**118 ± 26**	0.229	10.6
CLAHE	179 ± 48	0.213	10.7
CLAHE and SSC	185 ± 53	0.210	9.98
AOR Only	169 ± 36	0.135	10.4
Full Pipeline	176 ± 40	**0.073**	**3.13**

3.4.3 Summit XL Steel Sequences / Monocular VO

Pose Accuracy Comparison

The translation and orientation RMSE is reported in Table 3.5 for three different scenarios. The results show the accuracy enhancement in the case of the VO with the pipeline. For the Summit XL Steel robot sequences, a monocular VO algorithm was used for validating the proposed processing pipeline. Since the scale is unobservable, the wheel encoders of the robot are used to calculate its speed and the scale of the odometry. This means that a component of the error is due to the error in the velocity measurement taken by the encoders. However, since the same data is used for both cases, this effect is the same for both cases and will not make any bias in the comparison.

Table 3.5: Summit Accuracy Comparison

Seq. Name	VO without Pipeline		VO with Pipeline	
	Tr (m)	**Rot** (deg)	**Tr** (m)	**Rot** (deg)
Rectangle 1	0.49	1.67	**0.34**	**1.51**
Rectangle 2	0.25	1.69	**0.25**	**1.52**
Circle	0.50	2.09	**0.48**	**2.07**
Overall	0.31	1.75	**0.27**	**1.67**

Fig. 3.11 shows the estimation results for VO with and without the pipeline. As can be seen in the figure, the accuracy is superior in the case of the VO with the proposed pipeline. This is especially true at the end of the sequence, at which the VO without the pipeline suffered from large drift error. The presence of the AOR resulted in removing many matched outliers which would have caused bad motion estimation and a significant amount of drift errors in the case of VO without the proposed pipeline.

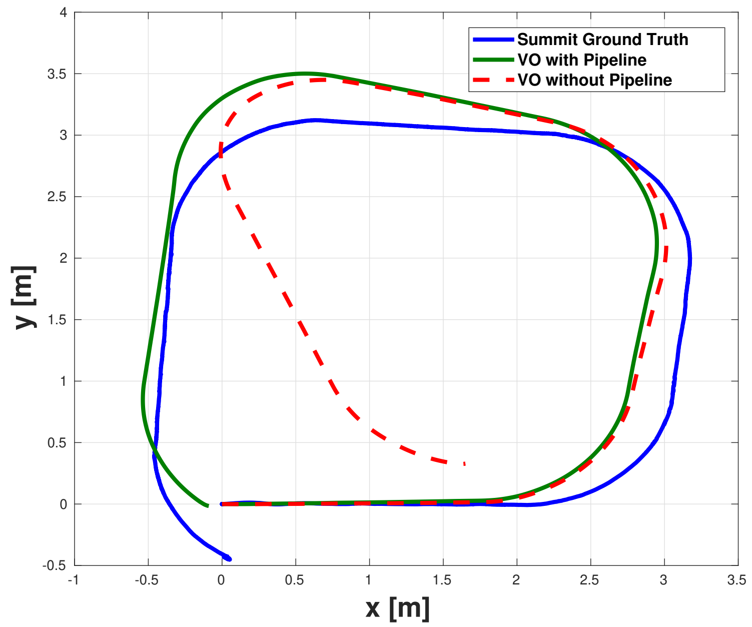


Figure 3.11: The pose estimation results for a rectangular sequence executed by Summit XLS steel. The Figure shows the output of the VO with and with the proposed pipeline. The VO with the pipeline shows better pose accuracy especially at the end of the sequence while the VO without the pipeline suffers from significant amount of drift error.

Computational Cost

Table 3.6 shows the computational time analysis of several combinations of VO algorithms including the proposed VO algorithm. As was illustrated before, the best computational performance was for the VO with the AOR algorithm. This is a direct result of the better outliers removal of the AOR which results in a better and faster convergence of the motion estimation algorithm. The Table also shows the translation and orientation RMSE for each of the different VO combinations. The results confirm that the proposed VO results in the best performance.

Table 3.6: Computation Time Comparison For Summit XLS Steel Sequences.

VO Type	Comp. Time (mean ± std [ms])	Tr RMSE (m)	Rot RMSE (deg)
Vanilla	148 ± 28	0.315	1.68
CLAHE	165 ± 47	0.350	1.68
CLAHE and SSC	168 ± 42	0.302	1.74
AOR Only	**132 ± 19**	0.287	1.68
Full Pipeline	158 ± 34	**0.275**	**1.67**

Table 3.7: The Pipeline Effect on VO

Dataset	Tr [%]	Rot [%]	Comp. Time [ms]
KITTI	-33%	-13%	+44
TUM	-68%	-70%	+58
Summit	-12%	-0.5%	+10

3.4.4 Discussion

For all the scenarios, and for all VO types used in this chapter, the proposed pipeline showed better performance compared to the VO without the pipeline. Specifically, adding the three additional stages to the actual VO algorithm enhanced the accuracy by an average of 37% for the considered datasets. These additional three stages can be added to any feature-based VO algorithm to enhance its accuracy and robustness.

In Table 3.2, 3.4 and 3.6, the results for different combinations of three proposed stages were reported. In the three cases, the VO with the full proposed pipeline showed better pose estimation accuracy. This shows that the three stages proposed in this chapter are integral. Each one of the three serves its own purpose and contributes to the overall enhancement. However, as expected, this came with an increase in the computational cost of the algorithm. Notice that the increase in the computational cost is still not significant (an average of 37 ms) and does not result in a large reduction in the number of frames per second. In most applications, this increase in the computational cost can be accepted, to improve the pose estimation accuracy in return (as shown in Table 3.7).

3.5 Conclusion

In this chapter, an image processing pipeline was introduced to enhance the accuracy and robustness of VO algorithms. The proposed pipeline consists of three stages, CLAHE, SSC, and AOR. Each stage addresses a separate issue associated with pose estimation error.

The proposed pipeline is intended to be generic and modular, which can be embedded in any feature-based algorithm in order to enhance its performance. In order to validate the proposed pipeline, sequences from KITTI and TUM datasets, as well as experimental sequences generated by a commercial omni-directional mobile robot were used. For each dataset, one type of VO was used for validation, namely stereo, RGB-D and monocular. The quantitative and qualitative results show that the proposed pipeline has a significant enhancement in the VO accuracy and robustness, with a minor increase in the computational time.

Chapter 4

Drift Covariance Estimation[1]

4.1 Introduction

In Chapter 3, a generic pipeline for enhancing VO output was introduced. However, as shown in the results of the chapter, the VO output still suffered from drift error. In fact, any type of odometry is bound to drift over time. Although several works in the literature attempt to reduce the drift in odometries (as discussed in Chapter 2.2.3), they still always suffer from drift due to their reliance on integration. Therefore, in this chapter, the estimation of the statistics of such drift is tackled. The proposed work can be integrated with any odometry in order to estimate the covariance and consequently, increase the accuracy of the localization module.

This chapter extends on the author's published work in [110]. In [110], the authors introduced the drift error model for wheel encoders and showed the feasibility of using an exteroceptive sensor to determine the covariance of a proprioceptive sensor, such as wheel encoders. In this chapter, the author generalizes the methodology and ideas adopted in [110] to any odometry, regardless of the type of sensor being used. The drift error model for any type of odometry and the drift covariance formula are derived. Furthermore, the estimation methodology introduced in [110] is generalized and more formally presented as the Drift Covariance Estimation (DCE) algorithm. The DCE is a generic algorithm, which can enhance the localization accuracy, when integrated to an odometry algorithm.

To show the benefits of using the DCE algorithm with popular open source odometry algorithms, the open source EU long-term dataset [169] is used, along with the LiDAR

[1]This chapter was published as Osman, Mostafa, et al., "A Novel Online Approach for Drift Covariance Estimation of Odometries Used in Intelligent Vehicle Localization," *Sensors*, 19.23 (2019): 5178 [111].

Odometry and Mapping algorithm (LOAM) [175]. The results of the DCE-UKF compared with the results of UKF with different constant covariance values are reported.

This chapter's main contributions are the following:

- The mathematical modeling of the random accumulation of error in the different odometries used in the localization of autonomous platforms.
- A general online algorithmic methodology for estimating the covariance of odometries using another sensor which is drift-free. This methodology is generic and suits any kind of odometry as well as any localization or SLAM algorithm either being filtering-based or optimization-based.
- The detailed implementation of an algorithm which follows this algorithmic methodology.

The remainder of the chapter is organized as follows; Section 4.2 derives the drift error model. Section 4.3 describes the proposed algorithm as well as some implementation details. Subsequently, in Section 4.4, the EU long-term dataset as well as the evaluation metrics used for validating the algorithm are introduced. Section 4.5 summarizes the results of the proposed algorithm, and finally in Section 4.6 the conclusion is summarized.

4.2 Drift Modeling in Odometries

Odometry and the drift error in odometry can be defined as

Definition (Odometry): Odometry is measuring the pose of a mobile platform $\mathbf{x} \in \mathbb{R}^n$ or a subset of it $\mathbf{z} \in \mathbb{R}^m$, by integrating the readings of a sensor after mapping it to the state (output) space,

$$\underbrace{C\mathbf{x}_k}_{\mathbf{z}_k} = \underbrace{C\mathbf{x}_{k-1}}_{\mathbf{z}_{k-1}} + \underbrace{\mathbf{g}(\mathbf{q})}_{\Delta\mathbf{z}_k}, \tag{4.1}$$

where $C \in \mathbb{B}^{m\times n}$ is the measurement matrix of the odometry, $\mathbb{B} := \{0, 1\}$ and for $C = \mathbf{I}_{n\times n}$, the odometry measures the overall state vector $\mathbf{x}$, $\mathbf{q} \in \mathbb{R}^s$ is the raw measurement of the sensor used in the odometry, and $\mathbf{g} : \mathbb{R}^s \to \mathbb{R}^m$ is the odometry mapping (maps the sensor measurements to the motion increments). In this chapter, the states of interest are only the position and orientation of the platform

$$\mathbf{x} := \begin{bmatrix} \mathbf{p} & \theta \end{bmatrix}^\top, \tag{4.2}$$

where $\mathbf{p} \in \mathbb{R}^3$ and $\theta \in \mathbb{R}^{n_\theta}$ are the position and orientation of the mobile platform, respectively and n_θ is the size fo the orientation vector depending on the representation used.

Definition (Drift Error): The drift error $\delta(\mathbf{z}, \epsilon, k) \in \mathbb{R}^m$ is the unbounded accumulation of error in the odometry measurements, due to the integration of sensor readings ,suffering from random noise $\epsilon \in \mathbb{R}^s$. For drift error, the following is true:

$$\delta(\mathbf{0}_{m\times 1}, 0) = \mathbf{0}_{m\times 1}, \tag{4.3a}$$

$$\lim_{\mathbf{z}\to\infty} \delta(\mathbf{z}, \cdot) = \infty, \qquad \text{and} \qquad \lim_{k\to\infty} \delta(\cdot, k) = \infty. \tag{4.3b}$$

Now the raw measurement $\mathbf{q}$ is related to the states through the sensor model mapping $\mathbf{h}_s : \mathbb{R}^n \to \mathbb{R}^s$ as

$$\mathbf{q}_k = \mathbf{h}_s(\mathbf{x}_k). \tag{4.4}$$

Eq. (4.4) can be written more precisely as a function of the observable states by the sensor $\mathbf{h}_s : \mathbb{R}^m \to \mathbb{R}^s$ as

$$\mathbf{q}_k = \mathbf{h}_s(\mathbf{C}\mathbf{x}_k) = \mathbf{h}_s(\mathbf{z}_k), \tag{4.5}$$

since the measurements of the sensor is only a function of the related states, which in this case is the $\mathbf{z}$ output vector.

Consequently, the ideal odometry equation will be

$$\mathbf{z}_k = \mathbf{z}_{k-1} + \mathbf{g}(\mathbf{q}_k) = \mathbf{z}_{k-1} + \mathbf{g}(\mathbf{h}_s(\mathbf{z}_k)). \tag{4.6}$$

Although the sensor measurement $\mathbf{h}_s$ is a function of the output vector $\mathbf{z}_k$, this is an implicit representation; the only way to measure the actual output vector $\mathbf{z}_k$ is through the odometry.

In reality, the readings $\mathbf{q}$ suffer from noises which can be represented as

$$\hat{\mathbf{q}} = \mathbf{h}_s(\mathbf{z}) + \epsilon, \tag{4.7}$$

where $\hat{\mathbf{q}}$ is the noisy readings of the sensors, and $\epsilon \in \mathbb{R}^s$ is a random variable representing the additive noise on the sensor measurements.

Substituting Eq. (4.7) in Eq. (4.6), the noisy odometry output can be written as

$$\hat{\mathbf{z}}_k = \hat{\mathbf{z}}_{k-1} + \mathbf{g}(\mathbf{q}_k + \epsilon_k). \tag{4.8}$$

Using the Taylor expansion, Eq. (4.8) can be written as

$$\hat{\mathbf{z}}_k = \hat{\mathbf{z}}_{k-1} + \mathbf{g}(\mathbf{q}_k) + \frac{\partial \mathbf{g}}{\partial \mathbf{q}_k}\epsilon_k + \text{Higher order terms,} \tag{4.9}$$

where

$$\frac{\partial \mathbf{g}}{\partial \mathbf{q}_k} =: \mathbf{J}_k \in \mathbb{R}^{m\times s},$$

is the Jacobian matrix.

Eq. (4.9) can also be rewritten as follows:

$$\hat{\mathbf{z}}_k = \sum_{i=0}^{k} \mathbf{z}_i + \underbrace{\sum_{i=0}^{k} \mathbf{J}_i\epsilon_i + \text{Higher order terms,}}_{\delta(\mathbf{z},\epsilon,k)} \tag{4.10}$$

The random nature of the drift error is a direct result of integrating the random noise ϵ.

Now assume a change $\Delta\mathbf{z}$ occurred in the pose over one timestep. Once again, through using the Taylor expansion, the drift error can be written as

$$\delta(\mathbf{z}+\Delta\mathbf{z},\epsilon,k) = \delta(\mathbf{z},\epsilon,k-1) + \frac{\partial\delta}{\partial\mathbf{z}}\Delta\mathbf{z} + \text{Higher order terms} \tag{4.11}$$

and the drift increment $\Delta\delta$ can be written as:

$$\Delta\delta = \underbrace{\frac{\partial\delta}{\partial\mathbf{z}}}_{\mathbf{K}_1}\Delta\mathbf{z} + \text{Higher order terms} \tag{4.12}$$

where $\mathbf{K}_1 \in \mathbb{R}^{m\times m}$ is the so-called drift coefficient matrix and is a random matrix which depends on the sensor random noise ϵ.

Since the sampling time is usually small, $\Delta\mathbf{z}$ is small and the higher order terms can be neglected.

$$\Delta\delta \approx \mathbf{K}_1\Delta\mathbf{z} \tag{4.13}$$

Furthermore, here we assume that the drift in each of the output states is independent from that of the other states, in other words,

$$\mathbf{K}_1 = \text{diag}\{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_m\}. \tag{4.14}$$

Using Eq. (4.13) in Eq. (4.11), the drift error at timestep k can be written as:

$$\delta_k = \delta_{k-1} + \Delta\delta_k \approx \delta_{k-1} + \mathbf{K}_1 \Delta\mathbf{z} \tag{4.15}$$

Now, the covariance of the drift increment $\mathcal{Q}_\delta \in \mathbb{R}^{m\times m}$ can be calculated as follows:

$$\begin{aligned}\mathcal{Q}_\delta &:= \mathbb{E}[(\mathbf{K}_1\Delta\mathbf{z} - \mu_\delta)(\mathbf{K}_1\Delta\mathbf{z} - \mu_\delta)^T] \\ &= \mathbb{E}[\mathbf{K}_1\Delta\mathbf{z}\Delta\mathbf{z}^T\mathbf{K}_1 - 2\mu_\delta\mathbf{K}_1\Delta\mathbf{z}^T + \mu_\delta\mu_\delta^T] \\ &= \mathbb{E}[\mathbf{K}_1\Delta\mathbf{z}\Delta\mathbf{z}^T\mathbf{K}_1] - \mu_\delta\mu_\delta^T,\end{aligned}$$

where $\mu_\delta \in \mathbb{R}^m$ is the mean of the drift increment.

Moreover, we assume that the drift increment is a small value due to the small time-step and therefore the square of its mean μ_δ is also small and can be neglected. Therefore, the drift increment covariance $\mathcal{Q}_\delta$ can be approximated as

$$\mathcal{Q}_{\delta_k} = \mathbb{E}[\mathbf{K}_{1,k}\Delta\mathbf{z}_k\Delta\mathbf{z}_k^T\mathbf{K}_{1,k}], \tag{4.16}$$

Using Eq. (4.13), the drift covariance matrix $\Sigma_\delta \in \mathbb{R}^{m\times m}$ can be calculated as

$$\Sigma_{\delta_k} = \Sigma_{\delta_{k-1}} + \mathcal{Q}_{\delta_k}. \tag{4.17}$$

Using Eq. (4.17), the drift covariance matrix Σ_{δ_k} can be estimated using a sensor which does not suffer from drift as will be shown in the following section.

4.3 Drift Covariance Estimation

Using the model developed in the previous section, the drift covariance Σ_δ can be estimated using a sensor which does not suffer from drift. Throughout the rest of the chapter, we refer to a sensor not suffering from drift as a drift-free sensor.

A good example for a drift-free sensor is a GPS, which does not rely on integration for measuring the pose of the mobile platform. Other examples of drift-free sensors are a camera detecting landmarks or apriltags, a magnetometer, and so on.

4.3.1 Drift Covariance Estimation Algorithm

To avoid notation ambiguity, in Section 4.3, a measurement from an odometry will be denoted $\hat{\mathbf{z}}_{\delta_k}$ and a drift-free sensor measurement will be denoted $\hat{\mathbf{z}}_k$.

Let us consider a drift-free sensor measurement $\hat{\mathbf{z}}_k \in \mathbb{R}^m$ with a known covariance $\mathcal{R}_k \in \mathbb{R}^{m\times m}$. The true output of the platform $\mathbf{z}_k$ is likely to lie in the confidence ellipse of the measurement, defined by the eigenvalues of $\mathcal{R}$. Since the true measurement $\mathbf{z}_k$ is not available, we resort to sampling the noisy measurement distribution $p(\hat{\mathbf{z}}_k|\mathbf{x}_k)$

$$\mathcal{Z}_k := \left\{\bar{\mathbf{z}}_{0_k} \quad \bar{\mathbf{z}}_{1_k} \quad \bar{\mathbf{z}}_{2_k} \quad \dots \quad \bar{\mathbf{z}}_{w_k}\right\}, \tag{4.18}$$

where $\mathcal{Z}_k \subset \mathbb{R}^m$ is the set of samples taken from the drift-free sensor distribution, w is the number of samples, and $\bar{\mathbf{z}}_{i_k}$ is the i-th sample in $\mathcal{Z}_k$ and $\bar{\mathbf{z}}_{0_k} = \hat{\mathbf{z}}_k$.

Using the samples in $\mathcal{Z}_k$, the innovations (error) between the odometry measurement and the samples are calculated as follows:

$$\lambda_{i_k} = \mathbf{e}_{\mathbf{z}}(\bar{\mathbf{z}}_{i_k}, \hat{\mathbf{z}}_{\delta_k}), \tag{4.19}$$

$$\Lambda_k := \left\{\lambda_{0_k} \quad \lambda_{1_k} \quad \dots \lambda_{w_k}\right\}, \tag{4.20}$$

where $\Lambda \subset \mathbb{R}^m$ is the set of innovation samples, and $\mathbf{e}_{\mathbf{z}} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is an error function, which can be defined as $\mathbf{e}_{\mathbf{z}}(\bar{\mathbf{z}}_{i_k}, \hat{\mathbf{z}}_{\delta_k}) := \bar{\mathbf{z}}_{i_k} - \hat{\mathbf{z}}_{\delta_k}$ in case of Cartesian states (position), but needs more elaborate design in case of orientations to calculate the innovation over the rotation manifold taking into account representation singularities. For more information, see [50, 24].

The innovation values are then stored in the so-called innovation memory tensor $\Xi \in \mathbb{R}^{n_v\times m\times w}$ which stores the last n_v innovation values, as shown in Eq. (4.21). Here, n_v is called the *measurement horizon*. The choice of the measurement horizon n_v is a design parameter; a trade-off between the accuracy of the covariance estimation and the computational cost.

$$\Xi = \begin{bmatrix}\Xi_0 & \Xi_1 & \dots & \Xi_w\end{bmatrix}^\top = \begin{bmatrix}\begin{bmatrix}\lambda^\top_{0_{k-n_v}} \\ \lambda^\top_{0_{k-n_v+1}} \\ \vdots \\ \lambda^\top_{0_k}\end{bmatrix} & \begin{bmatrix}\lambda^\top_{1_{k-n_v}} \\ \lambda^\top_{1_{k-n_v+1}} \\ \vdots \\ \lambda^\top_{1_k}\end{bmatrix} & \dots & \begin{bmatrix}\lambda^\top_{w_{k-n_v}} \\ \lambda^\top_{w_{k-n_v+1}} \\ \vdots \\ \lambda^\top_{w_k}\end{bmatrix}\end{bmatrix}^\top, \tag{4.21}$$

where $\Xi_i \in \mathbb{R}^{n_v\times m}, i \in \{0, \dots w\}$ is the matrix containing the n_v previous innovations calculated for the i-th track of sampled measurements.

Now let us recall the drift model in the previous section,

$$\Delta\delta = \mathbf{K}_1 \Delta\mathbf{z}, \qquad \text{and} \qquad \frac{\partial \delta}{\partial \mathbf{z}} = \mathbf{K}_1. \tag{4.22}$$

The drift coefficient $\mathbf{K}_1$ represents the increase in the drift error of the odometry with respect to the change in the output states $\mathbf{z}$. Therefore, using a drift-free sensor, the drift can be captured by estimating $\mathbf{K}_1$.

After we store the innovation points for n_v sample points, which encode the deviation between the odometry and several samples taken from the drift-free sensor. The slope of these values is an estimate of $\mathbf{K}_1$ since the normal progression of the output states $\mathbf{z}$ is captured by both sensors but only the drift suffering sensor has the drift component δ.

Using the innovation memory tensor Ξ, the drift coefficient $\mathbf{K}_1$ can be estimated for each track of the sample points, by computing the first order polynomial fit of the innovation data using a linear least squares approach, as shown in the following equations

$$\begin{bmatrix} \varrho \\ [\hat{\mathbf{K}}_1^i]^T \end{bmatrix} = (\bar{\mathbf{T}}^T \cdot \bar{\mathbf{T}})^{-1} \bar{\mathbf{T}}^T \boldsymbol{\Xi}_i, \tag{4.23}$$

$$\bar{\mathbf{T}} = \begin{bmatrix} \mathbf{1}_{n_v,1} & \mathbf{T} \end{bmatrix}, \tag{4.24}$$

$$[\hat{\mathbf{K}}_1^i]^T = \begin{bmatrix} \hat{\mathbf{k}}_1^i & \dots & \hat{\mathbf{k}}_m^i \end{bmatrix}, \tag{4.25}$$

where $\mathbf{T} \in \mathbb{R}^{n_v}$ is a vector containing the timestamps for the last n_v instances, $\hat{\mathbf{K}}_1^i \in \mathbb{R}^{m\times m}$ is the estimated drift coefficient matrix computed for Ξ_i, $[\hat{\mathbf{K}}_1^i] \in \mathbb{R}^m$ is a vector containing the diagonal values of the drift coefficient matrix, $\mathbf{1}_{n_v,1} \in \mathbb{R}^{n_v}$ is an all ones vector, and $\varrho \in \mathbb{R}^{1\times m}$ is the intercept which is of no use here. Eq. (4.23) is executed for each Ξ_i and w estimates of $\mathbf{K}_1$ are calculated.

Using these estimates, the drift increment covariance $\mathcal{Q}_\delta$ can be estimated using Eq. (4.16) as

$$\hat{\mathcal{Q}}_{\delta_k} = \mathbb{E}[\mathbf{K}_{1,k} \Delta\hat{\mathbf{z}}_{\delta_k} \Delta\hat{\mathbf{z}}_{\delta_k}^T \mathbf{K}_{1,k}] = \frac{1}{w} \sum_{i=0}^{w} \hat{\mathbf{K}}_{1,k}^i \Delta\hat{\mathbf{z}}_{\delta_k} \Delta\hat{\mathbf{z}}_{\delta_k}^T \hat{\mathbf{K}}_{1,k}^i. \tag{4.26}$$

Consequently, the drift covariance matrix can be calculated as

$$\hat{\Sigma}_{\delta_k} = \hat{\Sigma}_{\delta_{k-1}} + \mathbf{W} \hat{\mathcal{Q}}_{\delta_k}, \tag{4.27}$$

where $\mathbf{W} \in \mathbb{R}^{m\times m}$ is a diagonal wighting matrix. Notice that in Eq. (4.26), since the true output state increment $\Delta\mathbf{z}_k$ is not available, the increment from the odometry is used instead. To compensate for such change in the equation, a gain $\mathbf{W}$ is added to the algorithm. The covariance estimation algorithm is stated in Algorithm 4.

Algorithm 4: Drift Covariance Estimation Algorithm.

```
Initialize with: $n_v$ (measurement horizon) , $w$ (number of measurement samples), $\tau$ (sampling time)
for $k \in \{1, \ldots, \infty\}$ do
    for $i = 1 \ldots w$ do
        Sample $w$ sample points from the drift-free sensor measurement distribution, as shown in
          Eq. (4.18).
    end
    foreach $\bar{\mathbf{z}}_{i_k} \in \mathcal{Z}_k$ do
        Calculate the samples innovations as in Eq. (4.19)
        Push the innovation vector $\lambda_{i_k}$ to the innovation matrix $\Lambda_k$.
    end
    Pop the innovation matrix $\Lambda_{k-n_v-1}$ from the innovation memory tensor $\Xi$ and the timestep
      $k - n_v - 1$ from the time vector $\mathbf{T}$.
    Push the innovation matrix $\Lambda_k$ to the innovation memory tensor $\Xi$ and the timestep $k$ to the
      time vector $\mathbf{T}$.
    for $i \in \{1 \ldots w\}$ do
        Calculate the estimated drift coefficients using a first order polynomial fit of the
          innovation data as in Eq. (4.23).
    end
    Calculate the estimated drift increment covariance and the drift covariance, as shown in
      Eq. (4.26) and Eq. (4.27), respectively.
end
```

4.3.2 Algorithm Implementation

A generic implementation of the covariance estimation algorithm is developed and available as an online open-source repository in [107]. This implementation uses the unscented transform introduced in [60, 158]. Normally, the unscented transform is used in the UKF; however, it also fits for the proposed covariance estimation algorithm, for sampling the drift-free measurement distribution.

Using the unscented transform, the drift-free sensor distribution is sampled as

$$\mathcal{Z}_k = \left[\hat{\mathbf{z}}_k \quad \hat{\mathbf{z}}_k + (\upsilon\sqrt{\mathcal{R}_k}) \quad \hat{\mathbf{z}}_k - (\upsilon\sqrt{\mathcal{R}_k})\right] \in \mathbb{R}^{m\times 2m+1} \tag{4.28}$$

where $\upsilon \in \mathbb{R}_{>0}$ is the sampling parameter in the unscented transform.

The υ parameter is normally calculated as $\upsilon = m + \nu$, where ν is a design parameter (see [158], for more information). However, here we directly specify υ itself, because it does not affect the performance of the DCE.

The choice of the parameter υ depends on the accuracy of the drift-free sensor covariance. More specifically, if the drift-free sensor covariance $\mathcal{R}_k$ reflects under-confidence in the sensor (the sensor is more accurate than reflected in the covariance), then the true measurement $\mathbf{z}$ is more likely to be in the confidence ellipse and consequently, υ should be less than or equal to 1. However, if the covariance reflects over-confidence, then υ should be greater than 1.

Afterwards, instead of calculating the mean of the drift coefficients, we calculate a weighted mean as

$$W_0 = \frac{m}{2m+1} \text{ and } W_i = \frac{m+1}{4m^2+2m}, \tag{4.29}$$

$$\hat{\mathcal{Q}}_{\delta_k} = \sum_{i=0}^{2m} W_i \hat{\mathbf{K}}^i_{1,k} \Delta\hat{\mathbf{z}}_{\delta_k} \Delta\hat{\mathbf{z}}^T_{\delta_k} \hat{\mathbf{K}}^i_{1,k}, \tag{4.30}$$

where $W_i \in \mathbb{R}_{>0}$ is the weight for a given sampled measurement.

The choice of the weights calculation in Eq. (4.29) is a design choice. Here, we choose to give a higher weight to the sensor measurement $\hat{\mathbf{z}}$, while giving equal weights to all other samples. Again, this is a design parameter which can be changed or even completely removed from the algorithm implementation.

Notice that the proposed algorithm only estimates the covariance of the odometry output $\hat{\mathbf{z}}_\delta$ but does not correct such drift. The correction can be done through using the output of the localization algorithm $\hat{\mathbf{x}}$ as shown in Eq. 4.31 and Eq. 4.32 and illustrated in Fig. 4.1.

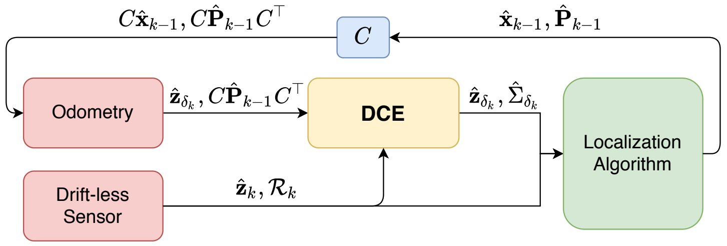

$C\hat{\mathbf{x}}_{k-1}, C\hat{\mathbf{P}}_{k-1}C^\top$
C
$\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}$
Odometry
$\hat{\mathbf{z}}_{\delta_k}, C\hat{\mathbf{P}}_{k-1}C^\top$
DCE
$\hat{\mathbf{z}}_{\delta_k}, \hat{\Sigma}_{\delta_k}$
Localization Algorithm
Drift-less Sensor
$\hat{\mathbf{z}}_k, \mathcal{R}_k$

Figure 4.1: The structure of the localization system with the DCE algorithm. The drift is corrected using the feedback from the localization algorithm.

$$\hat{\mathbf{z}}_{\delta_k} = C\hat{\mathbf{x}}_{k-1} + \Delta\hat{\mathbf{z}}_{\delta_k} \tag{4.31}$$

$$\hat{\Sigma}_{\delta_k} = C\hat{\mathbf{P}}_{k-1}C^\top + \mathbf{W}\hat{\mathcal{Q}}_{\delta_k} \tag{4.32}$$

where $\mathbf{W} \in \mathbb{R}^{m \times m}$ is a tuning parameter to avoid under-confidence in the estimated covariance. However, in all the experiments, this matrix was set to $\mathbf{I}_{m \times m}$.

Through using the drift covariance estimation, the covariance of the odometry can be accurately estimated. The odometry measurement with its estimated covariance are then provided to the localization algorithm. This will help correct the drift in the odometry during localization.

Time and Memory Complexity

In this subsection, the time and memory complexities of the algorithm are discussed. The most computationally expensive operation, in the proposed implementation, is the sampling of the measurement distribution due to the matrix square root operation. The matrix square root is typically performed using Cholesky decomposition. In general, the time complexity of the Cholesky decomposition is $\mathrm{O}(m^3)$ [71]. Although this is normally computationally expensive, in this case, the worst-case scenario covariance matrix is a 6×6 matrix which is a small matrix size. Furthermore, if the noise in the sensor readings is uncorrelated (for different states), the covariance matrix reduces to a diagonal matrix and the matrix square root reduces to a set of normal square root operations which is a constant time operation. Consequently, ignoring the matrix square root, several steps of the algorithm iterates over each sampled measurement, making the algorithm complexity $\mathrm{O}(w)$. Notice that the matrix inversion of $(\bar{\mathbf{T}}^\top \cdot \bar{\mathbf{T}})$ is only an inversion operation of a 2×2 matrix which is a constant time operation.

As for the memory size allocation, this can be specified using the innovation memory tensor. In each iteration of the algorithm, the innovation memory tensor must store the m measured states for each of the w samples generated for each of the time-steps in the measurement horizon n_v. This makes the memory complexity of the algorithm $\mathrm{O}(m \cdot w \cdot n_v)$.

4.4 Experimental Work

To validate the DCE algorithm, the results of using an open source implementation of the LOAM algorithm in a UKF multi-sensor fusion localization are reported. The EU long-term dataset [169] was chosen for the validation, and the accuracy of the DCE-UKF algorithm is compared to different constant covariance values. This section describes the EU long-term dataset as well as the evaluation metrics used for validating the performance of the proposed DCE (see [111] for more testing case studies).

4.4.1 EU Long-Term Dataset

The EU long-term Dataset is a dataset for autonomous driving containing the data from multiple heterogeneous sensors. The data was collected using the University of Technology of Belfort Montbliard (UTBM) vehicle, in human driving mode, by driving the vehicle in the downtown of Montpelier in France [169]. For the long-term data, the driving distance is about 5.0 km per session driven in 16 minutes. Here, three driving data sequences from the dataset are used to validate the proposed DCE algorithm; all of the sequences are for the same path shown in Fig. 4.2.

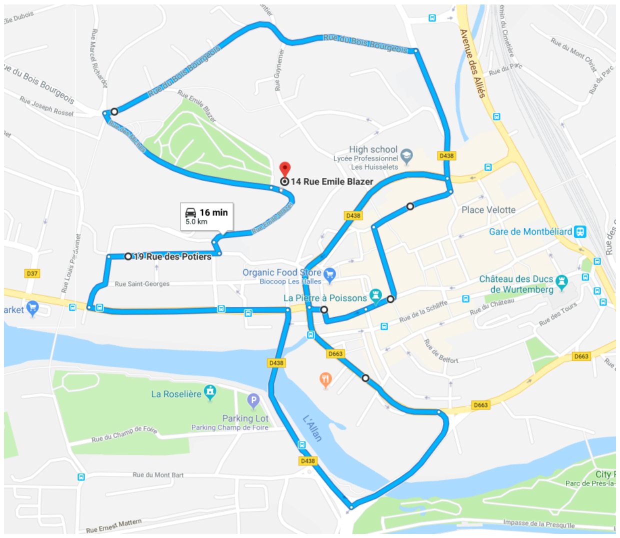


Figure 4.2: The path of the vehicle in UTBM dataset sequences.

The ground-truth for this dataset is generated by a GPS/RTK sensor which can achieve very accurate position measurements [56]. Therefore, an artificial Gaussian noise $\mathcal{N}(0, \mathcal{Q}_{gps})$ with $\mathcal{Q}_{gps} := \text{diag}(25[\text{m}^2], 25[\text{m}^2], 25[\text{deg}^2])$ is added to the GPS data and used as the drift-free sensor measurements. Furthermore, an open source implementation of the the LiDAR Odometry and Mapping (LOAM) algorithm [175] is used as the odometry measurements.

4.4.2 Evaluation Metrics

The DCE-UKF is evaluated using translation and orientation metrics. First, the mean and maximum error percentages of the translation are calculated as shown in Eq. (4.33) and

Eq. (4.34) respectively.

$$TE_{mean}[\%] = \frac{1}{L \cdot D} \sum_{k=1}^{N} \|\hat{\mathbf{p}}_k - \mathbf{p}_k\|_2 \,, \tag{4.33}$$

$$TE_{max}[\%] = \frac{1}{D} \cdot \max_{k \in \mathbb{N}_0} \|\hat{\mathbf{p}}_k - \mathbf{p}_k\|_2 \,, \tag{4.34}$$

where $\hat{\mathbf{p}}_k$ and $\mathbf{p}_k$ are the estimated and true position at time-step k respectively, L is the total number of estimated poses in the driving sequence, and D is the total distance of the driving sequence.

Second, the mean and maximum errors of the orientation, as a ratio of the overall distance of the sequence, are calculated as shown in Eq. (4.35) and Eq. (4.36), respectively.

$$OE_{mean}[^o/m] = \frac{1}{L \cdot D} \sum_{k=1}^{N} \|\hat{\theta}_k - \theta_k\|_1, \tag{4.35}$$

$$OE_{max}[^o/m] = \frac{1}{D} \cdot \max_{k \in \mathbb{N}_0} \|\hat{\theta}_k - \theta_k\|_1, \tag{4.36}$$

where $\hat{\theta}_k$ and θ_k are the estimated and true orientation of the vehicle at time-step k respectively. The true states of the vehicle (x, y) coordinates are obtained using the RTK-GPS sensor. The yaw heading angle of the vehicle is calculated as the slope of the (x, y) coordinates measured by the RTK-GPS.

4.5 Results and Discussion

A total of three sessions (2018-05-02 (1), 2018-05-02 (2), 2018-07-13) of EU long-term dataset were used for validating the DCE algorithm. These are a total of 15,000 m of driving distance divided over three days in different weather conditions.

To show the efficacy of the DCE algorithm, its localization results are compared to the results of different values of constant covariances. The values used are percentages of the True Variance (TV) of the odometries, calculated retroactively from the data of the experiments.

The covariances of the LOAM odometry used for comparison are calculated using the TV along with the correction step as described in Fig. 4.1 as follows

$$\hat{\Sigma}_k = C\hat{\mathbf{P}}_{k-1}C^\top + \mathcal{Q}_{TV}\;[\Delta \mathbf{z}_{\delta_k}], \tag{4.37}$$

where $\mathcal{Q}_{TV} \in \mathbb{R}^{m\times m}$ is the diagonal covariance matrix containing the TV for each of the output states, and $[\Delta \mathbf{z}_{\delta_k}] \in \mathbb{R}^{m\times m}$ is the motion increment vector in a diagonal matrix form.

Table 4.1 shows the results of the DCE-UKF algorithm compared with different constant covariances. As shown in the table, the DCE-UKF outperforms the UKF with constant covariances. The results show that the DCE algorithm can result in enhancing the accuracy of the pose estimation when integrated with an odometry algorithm.

Table 4.1: Mean of the overall EU Dataset UKF results

Metrics	**Mean** [%]			
	$\boldsymbol{TE_{mean}}$	$\boldsymbol{TE_{max}}$	$\boldsymbol{OE_{mean}}$	$\boldsymbol{OE_{max}}$
DCE-UKF	**0.0694**	**0.2757**	**0.0007**	**0.0031**
True Variance	0.378	1.579	0.0059	0.0789
2.5% TV	0.075	0.297	0.0014	0.0068
5% TV	0.0824	0.357	0.0017	0.0097
25% TV	0.158	2.014	0.0037	0.0246
125% TV	0.395	2.782	0.0098	0.0568

The data in the EU long-term dataset was recorded while the vehicle was moving with a relatively high speed, consequently, the LOAM algorithm suffered from a substantial amount of drift error. Using the DCE algorithm, a much better pose estimation was achieved through sensor fusion. Fig. 4.3 shows the estimated trajectory using DCE-UKF compared to the results of the LOAM alone, which suffered from substantial amount of drift as shown in the figure. In addition to the LOAM data, the figure also shows the noisy data of the GPS.

Table 4.2 shows the resulted translation and orientation metrics for the DCE-UKF, LOAM, GPS and filtered GPS. The use of the DCE-UKF along with the LOAM data achieved much better results than just using the LOAM. This shows the importance of using a fusion algorithm with accurate covariances to achieve more accurate pose estimation.

Due to the high amount of drift error in the LOAM, as shown in Fig. 4.3, one might think that only filtering the GPS readings by the UKF might lead to a better results than fusing the GPS with a LiDAR odometry and using the DCE algorithm. For this, we also show the results of filtered GPS readings using a UKF. As can be seen in Table 4.2, the result of fusing LOAM with the noisy GPS readings led to better results than ignoring the data from the LOAM completely.

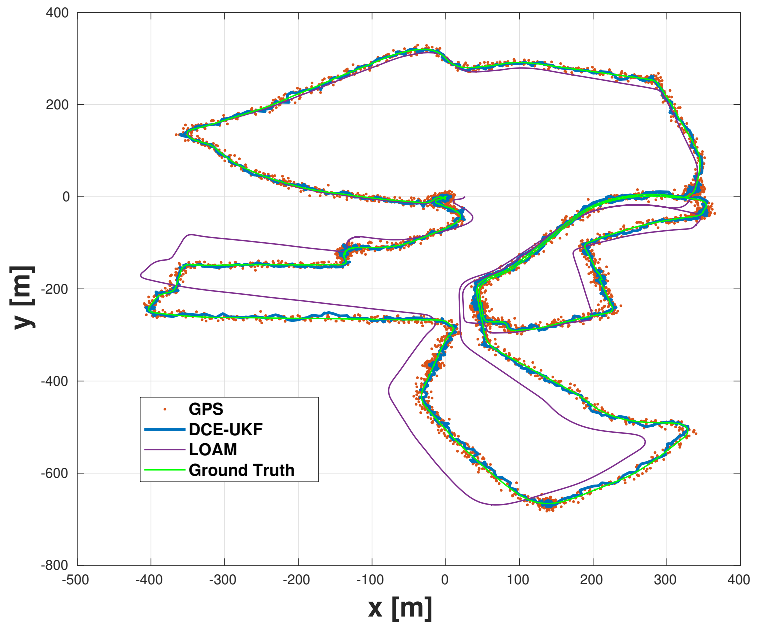


Figure 4.3: The EU dataset results of the DCE-UKF, LOAM, and GPS for sequence 2018-07-13. Note: The results shown in [111] were collected while running LOAM algorithm in real-time. However, the results shown in this thesis were collected using pre-recorded LOAM measurements recorded at a slower speed in order to get better measurements from the LOAM algorithm.

Finally, all the experiments were executed using an Intel Core i7 CPU at 2.10 GHz processor. The average computation time of the algorithm was 13.3 ms (for $n_v = 20$). This small computation time indicates the feasibility of implementing the proposed DCE algorithm in real-time for localization of vehicles. Furthermore, the maximum and minimum computation time were 23.3 ms and 8.1 ms respectively along with standard deviation of 3.5 ms, which show that the performance of the DCE algorithm is consistent and does not significantly fluctuate through operation.

4.6 Conclusion

This chapter presented a novel approach for estimating the covariance of odometries suffering from drift due integration. The algorithm uses the covariance of another sensor, which

Table 4.2: The EU dataset results of the DCE-UKF, LOAM, GPS and filtered GPS for sequences 2018-05-02 (1, 2) and 2018-07-13.

Metrics	**Mean [%]**			
	$\boldsymbol{TE_{mean}}$	$\boldsymbol{TE_{max}}$	$\boldsymbol{OE_{mean}}$	$\boldsymbol{OE_{max}}$
DCE-UKF	**0.0694**	**0.278**	**0.0007**	**0.0031**
LOAM	0.467	1.393	0.0067	0.0580
GPS	0.106	0.380	0.0015	0.0044
Filtered GPS	0.082	0.319	0.0009	0.0040

does not suffer from drift (drift-free sensor). The drift covariance estimation algorithm overcomes the challenges of quantifying the constant covariances through the presence of the ground-truth or through hard-tuning. Furthermore, DCE takes into consideration the fact that the covariance of such odometries is dynamic and changes with time during the operation of the sensor.

The drift covariance estimation algorithm was tested using the EU Long-term Dataset and UKF. The localization output while using the DCE was compared to the localization output with different constant covariance values. The results showed that the DCE algorithm outperformed the constant covariances in all the experiments which confirms that the use of constant covariances, for drift suffering sensors, is neither optimal nor practical.

Chapter 5

Moving Horizon Estimation Localization[1]

5.1 Introduction

In Chapter 3, we showed that a visual odometry can be enhanced through the use of several image processing techniques. Afterwards, in chapter 4, we introduced a generic covariance estimation scheme referred to as the Drift Covariance Estimation (DCE) to estimate the uncertainty in drift suffering odometries. Moreover, in Section 2.2.4 and Section 2.2.5, we described the MHE and how does it relate to MAP and Bayesian estimation techniques. In this chapter, we utilize the MHE formulation to develop a multi-sensor fusion scheme for accurate localization of autonomous platforms.

In this chapter, a generic multi-sensor fusion framework is developed for the localization of autonomous platforms using MHE. The proposed method is based on a multi-threading architecture to meet the computational requirements for practical deployment. The MHE fusion scheme is tested using both simulated data as well as experimental data sequences. The MHE estimation output is compared to that of a UKF. Finally, an implementation of the proposed estimation scheme is made open-source for the benefit of the scientific community. The main contributions of this chapter are:

- The development of a generic multi-sensor fusion scheme based on MHE for the

[1]This chapter is under submission at *Information Fusion, Elsevier* as Mostafa Osman, Mohamed W. Mehrez, Mohamed A. Daoud, Ahmed Hussein, Soo Jeon, and William Melek, "A Generic Multi-sensor Fusion Scheme for Autonomous Platforms' Localization Using Moving Horizon Estimation.".

localization of autonomous systems using any number of sensors. The generic fusion scheme considers different sensor rates as well as missing measurements and outliers.

- A threaded realization of the MHE algorithm to reduce its computational cost and enable deployment in practical applications.
- A ROS-based open-source implementation of the proposed methodology to facilitate its validation.

The remainder of this chapter is organized as follows: in Section 5.2, the generic fusion scheme developed for fusing different odometries using MHE is presented. In Section 5.4, the MHE localization results are compared with the results of the UKF multi-sensor fusion algorithm. Finally, concluding remarks are discussed in Section 5.5.

5.2 Multi-sensor Fusion Using MHE for Localization

In this section, a generic multi-sensor fusion scheme based on MHE (see Section 2.2.5 for more information) is proposed for the localization of autonomous platforms; the localization algorithm considers any arbitrary number of odometries, e.g. LiDAR, wheel encoders, and IMU, as well as global measurements such as measurements from a GPS. The generic motion model used for fusion is presented followed by the measurement model and the specific (measurement) data handling and processing methodology. Finally, the proposed multi-threading fusion algorithm is introduced.

5.2.1 Motion Model

An autonomous platform (ground, underwater, or aerial) states can be defined as

$$\mathbf{x} := \begin{bmatrix}\mathbf{p} & \theta & \dot{\mathbf{p}} & \omega & \mathbf{a}\end{bmatrix}^\top, \tag{5.1}$$

where $\mathbf{p} := [x, y, z]^\top \in \mathbb{R}^3$ is the position of the platform in Cartesian coordinates, $\theta := [\alpha, \beta, \gamma]^\top \in \mathbb{R}^3$ is the orientation of the robot represented in Euler angles representation [89, Chapter 3], $\dot{\mathbf{p}} := [v_x, v_y, v_z]^\top \in \mathbb{R}^3$, $\omega := [\omega_x, \omega_y, \omega_z]^\top \in \mathbb{R}^3$, and $\mathbf{a} := [a_x, a_y, a_z]^\top \in \mathbb{R}^3$ are the linear velocity, angular velocity and linear acceleration vectors of the robot relative to the body frame, respectively.

For $k \in \mathbb{N}$, the following generic 3D omni-directional motion model is used as the prediction model for the fusion scheme [138, Chapter 3]

$$\begin{bmatrix} \mathbf{p}_{k+1} \\ \theta_{k+1} \\ \dot{\mathbf{p}}_{k+1} \\ \omega_{k+1} \\ \mathbf{a}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \tau R_k & \mathbf{0}_{3\times3} & \frac{\tau^2}{2} R_k \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \tau \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \tau \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{k} \\ \theta_{k} \\ \dot{\mathbf{p}}_{k} \\ \omega_{k} \\ \mathbf{a}_{k} \end{bmatrix}, \tag{5.2}$$

where $\mathbf{0}_{3\times3}$ and $\mathbf{I}_{3\times3}$ are 3×3 zero and identity matrices with respective dimensions defined by the subscript, and $R_k := R(\alpha_k, \beta_k, \gamma_k) \in SO(3)$ is the 3D rotation matrix of the robot relative to the world frame, where $SO(3)$ is the special orthogonal group [89, Chapter 3].

5.2.2 Measurements Model

An autonomous platform can be localized by using sensors such as LiDARs, cameras, wheel encoders, IMUs, and GPSs or a combination of some or all of the above. The localization can be accomplished by odometry computation (odometry-based), e.g. LiDAR, visual or wheel odometries, through map-based localization, e.g. landmark-based and grid-based localization [147], or using exteroceptive sensor such as GPS or magnetometer [138].

In this thesis, we assume that each of the robot's on-board sensors provides a measurement for a subset of the platform's states $\mathbf{x}$; thus, the measurement model of each sensor is linear as follows

$$\mathbf{z}_k^j = C^j \mathbf{x}_k^j, \tag{5.3}$$

where $j \in \{1, \dots M\}$ is the sensor index, M is the number of sensors used for localization, $C^j \in \mathbb{B}^{m_j \times n}$ is the measurement matrix of the j-th sensor, where $\mathbb{B} := \{0, 1\}$.

Assuming linear measurement models is acceptable, because most of the sensors used in the localization of autonomous platforms, in general, provide a subset of the vehicle states. For example, a visual or LiDAR odometry of a ground mobile robot provides a measurement of $[x, y, \gamma]$, a barometer can be used to provide the altitude z of a quad-copter, and a gyroscope can provide a measurement of the angles $[\alpha, \beta, \gamma]$.

5.2.3 Error Definition over the Rotation Manifold

As explained in Section 2.2.5, the MHE cost function (2.30) embodies the error in the model and the measurement. The error vector can be computed for the position, velocity, angular velocity and acceleration in the Cartesian space as

$$\begin{aligned}
\mathbf{e}_{\mathbf{p}}(\mathbf{p}, \hat{\mathbf{p}}) &:= \mathbf{p} - \hat{\mathbf{p}} \in \mathbb{R}^3, \\
\mathbf{e}_{\mathbf{v}}(\mathbf{v}, \hat{\mathbf{v}}) &:= \mathbf{v} - \hat{\mathbf{v}} \in \mathbb{R}^3, \\
\mathbf{e}_{\omega}(\omega, \hat{\omega}) &:= \omega - \hat{\omega} \in \mathbb{R}^3, \\
\mathbf{e}_{\mathbf{a}}(\mathbf{a}, \hat{\mathbf{a}}) &:= \mathbf{a} - \hat{\mathbf{a}} \in \mathbb{R}^3.
\end{aligned}$$

Such error definition can be used when the states are a subset of an Euclidean space. However, rotations are not completely defined over the Euclidean space and the Euler angles representation contains singularities. Therefore, in order to globally define the error function for rotations, the special orthogonal group $SO(3)$ is used to define the orientation error as follows [24]

$$\mathbf{e}_{\theta}(\theta, \hat{\theta}) := \begin{bmatrix} \tilde{r}_{32} - \tilde{r}_{23} \\ \tilde{r}_{13} - \tilde{r}_{31} \\ \tilde{r}_{21} - \tilde{r}_{12} \end{bmatrix} \in \mathbb{R}^3,$$

where $\tilde{r}_{ij}$ is the i-th row and j-th column element of the error rotation matrix $\tilde{R}$ defined as

$$\tilde{R} := R(\alpha_k, \beta_k, \gamma_k) R^{\top}(\hat{\alpha}_k, \hat{\beta}_k, \hat{\gamma}_k).$$

Finally, the error vector of the states $\mathbf{e}_{\mathbf{x}}$ can be defined as

$$\mathbf{e}_{\mathbf{x}}(\mathbf{x}, \hat{\mathbf{x}}) := \begin{bmatrix} \mathbf{e}_{\mathbf{p}}(\mathbf{p}, \hat{\mathbf{p}}) \\ \mathbf{e}_{\theta}(\theta, \hat{\theta}) \\ \mathbf{e}_{\mathbf{v}}(\mathbf{v}, \hat{\mathbf{v}}) \\ \mathbf{e}_{\omega}(\omega, \hat{\omega}) \\ \mathbf{e}_{\mathbf{a}}(\mathbf{a}, \hat{\mathbf{a}}) \end{bmatrix} \in \mathbb{R}^{15}, \tag{5.4}$$

and the measurements error for each sensor can be defined as a subset of the states error vector as

$$\mathbf{e}^j_{\mathbf{z}}(\mathbf{z}, \hat{\mathbf{z}}) := C^j \mathbf{e}_{\mathbf{x}}(\mathbf{x}, \hat{\mathbf{x}}). \tag{5.5}$$

5.2.4 Multi-sensor Fusion Scheme

In order to develop a generic framework for the MHE localization, the following requirements need to be addressed:

1. The localization algorithm has to consider the different measurement rates of the used sensors as well as the missed measurements instances.
2. The localization algorithm has to meet the practical application requirements, i.e. the MHE optimization problem (2.32) has to be solved in real-time for the fusion scheme to be practical; hence, a multi-threading architecture is implemented to lower the computational cost of the multi-sensor fusion scheme.
3. The motion model is generic; however, certain kinematic constraints need to be specified for such a model depending on the type of platform, e.g. holonomic, non-holonomic.
4. An outlier rejection method has to be implemented to prevent outliers from being included in the optimization as this can significantly deteriorate the fusion performance.

Each of the aforementioned considerations is addressed through the proposed multi-sensor fusion scheme as follows.

Multi-rate Multi-sensor Fusion

The proposed localization algorithm does not impose any limits on the update rates of the sensors being fused. Here, the localization scheme relies on solving the optimization problem (2.32) using a fixed but adjustable sampling rate. The different sensor measurements are received at different times and depending on the measurement received during the sampling time duration, a new iteration of the MHE is formulated using the available data, see Fig. 5.1 for an illustration of the proposed concept. The different queues of the data from the sensors (first-in-first-out) over the estimation horizon are updated with the new measurements; then, the measurements older than the starting instance of the present estimation horizon are discarded. The cost function (2.30) is then updated by the new available measurements.

Multi-Threading Architecture

The computational cost of solving the MHE problem is highly dependent on the estimation horizon length N. Nonetheless, some considerations can be taken to reduce the computational cost of the data handling as well as the computation of the information matrices (the inverse of the covariance matrix). Consequently, in order to keep the computational cost as low as possible given the estimation horizon N, the proposed localization scheme is

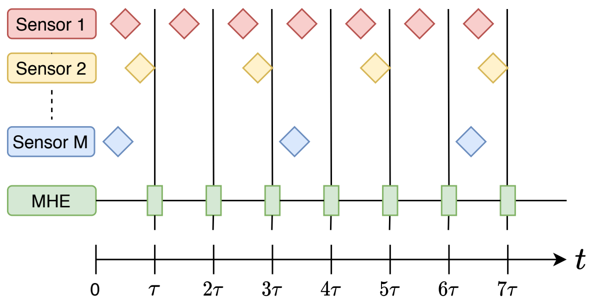


Figure 5.1: Multirate MHE-based localization, where M is the number of sensor whose measurements are being fused. Diamond shapes represents the available time-stamped measurements produced from each sensor.

developed in a multi-threading architecture; here, the measurement data is received and handled in a separate thread (front-end) from the main thread (back-end) used for solving the MHE optimization problem (2.32), see Fig. 5.2.

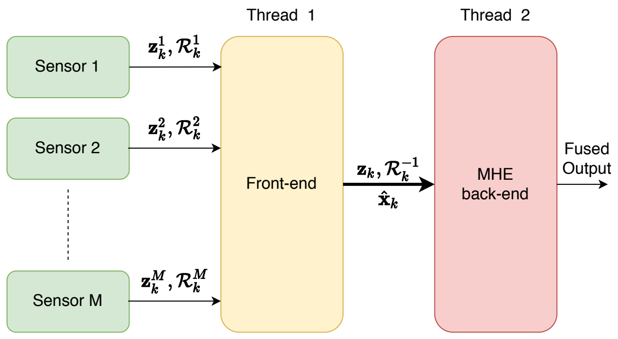

Thread 1
Thread 2
Sensor 1
Sensor 2
Sensor M
$\mathbf{z}_k^1, \mathcal{R}_k^1$
$\mathbf{z}_k^2, \mathcal{R}_k^2$
$\mathbf{z}_k^M, \mathcal{R}_k^M$
Front-end
$\mathbf{z}_k, \mathcal{R}_k^{-1}$
$\hat{\mathbf{x}}_k$
MHE back-end
Fused Output

Figure 5.2: The multi-threading architecture of the MHE localization scheme.

Thread 1 is responsible for receiving the measurements from the different sensors, computing the information matrices for each of the sensors, building the concatenated measurement vector, information matrix, and measurement matrix, i.e.

$$\mathbf{z}_k := \begin{bmatrix} \mathbf{z}_k^1 \\ \vdots \\ \mathbf{z}_k^M \end{bmatrix}, C_k := \begin{bmatrix} C_k^1 \\ \vdots \\ C_k^M \end{bmatrix}, \text{and } \ \mathcal{R}_k := \begin{bmatrix} \mathcal{R}_k^1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathcal{R}_k^M \end{bmatrix},$$

as well as propagate the motion model through the next time-step. Finally, the measurement vector $\mathbf{z}_k$ and the information matrix $\mathcal{R}_k^{-1}$ are pushed into the estimation horizon queue and the oldest measurement in the queue is discarded while only keeping the posterior covariance (calculated using Eq. (2.31)) for the arrival cost calculation. Moreover, the the posterior covariance is calculated. In Thread 2, the optimization problem (2.32) is updated and solved.

This architecture reduces the computational cost of the MHE through dividing the required computations into two threads, which can work in parallel. The first thread receives the measurement, computes the new prediction, inverts the covariance matrices, and forms the measurements vector while the other thread is solving the previous optimization problem. This results in a reduction in the overall computational time of the MHE and increases lower bound on the sampling time as shown in Fig. 5.3.

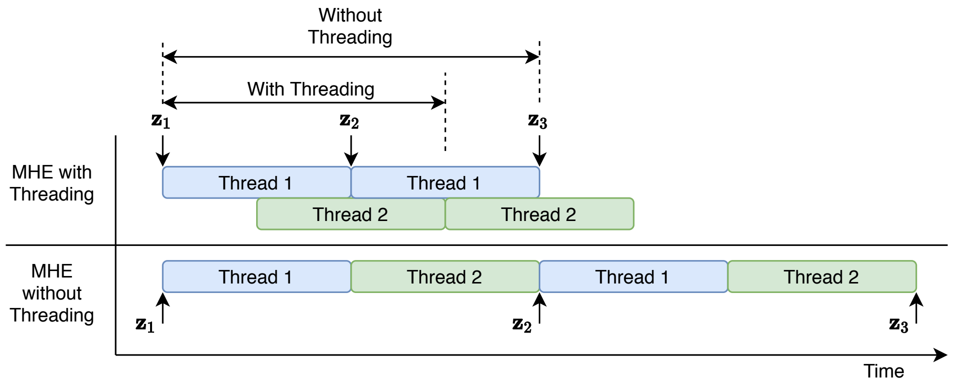


Figure 5.3: Visualization of the computational time for the MHE scheme in the multi-threading case as well as the unthreaded case. The computational time decrease in case of the multi-threading case as new measurements can be received and processed before the previous optimization problem is solved.

Compensation for the Vehicle Type

The proposed MHE localization scheme relies on a generic omni-directional motion model (Eq. (5.2)) for state prediction. Therefore, when solving the MHE optimization problem, certain kinematic constraint need to be included into the optimization problem to compensate for the vehicle type. For example, a differential mobile robot has a constraint on lateral motion, i.e. $v_y = 0$. Such constraint is not directly modeled in the aforementioned motion model, but it can be explicitly imposed on the optimization problem.

Outlier Rejection

Measurement outliers are detected and removed using the Mahalanobis distance thresholding method [91] as follows:

$$\mathcal{E}_k^j = \mathbf{e}_\mathbf{z}^\top(\mathbf{z}_k^j, \hat{\mathbf{z}}_k^j)\mathcal{R}^{-1}\mathbf{e}_\mathbf{z}(\mathbf{z}_k^j, \hat{\mathbf{z}}_k^j) \leq \mathcal{G}^j,\ j \in \{1, \dots M\} \tag{5.6}$$

where $\mathcal{E}_k^j$ is the Mahalanobis distance, and $\mathcal{G}^j \in \mathbb{R}_{>0}$ is the configurable Mahalanobis threshold. Algorithm 5 shows the proposed MHE-based localization framework and a ROS-based implementation is available online in [106].

Algorithm 5: MHE generic multi-sensor fusion scheme for autonomous vehicles localization

```
Set The estimation horizon $N$, number of sensors $M$, process covariance $\mathcal{Q}$, sampling time $\tau$,
  measurement matrices $C^i$, and the threshold $\mathcal{G}^i$ for $i \in \{1, \dots, M\}$
Set the optimization constraints.
for $k \in \{1, ..., \infty\}$ do
    Thread 1:
    {
    Propagate the states $\mathbf{x}_{k-1}$ through the motion model.
    for $i \in \{1, \dots, M\}$ do
        Receive the available sensors' measurements and covariance matrices $\mathbf{z}_k^i, \mathcal{R}_k^i$.
        Calculate the information matrix ${\mathcal{R}_k^i}^{-1}$ for each measurement.
        Calculate the Mahalanobis distance $\mathcal{E}_i$ for each measurement via Eq. (5.6).
        if $\mathcal{E}_k^i < \mathcal{G}^i$ then
            Push the sensor measurement, and the information matrix to $\mathbf{z}_k$ and $\mathcal{R}_k^{-1}$.
        end
    end
    Push the measurement vector $\mathbf{z}_k$, and information matrix $\mathcal{R}_k^{-1}$ to the estimation horizon queue
      and remove the oldest measurement from the queue.
    Calculate the posterior covariance using Eq. (2.31).
    }
    Thread 2:
    {
    Update the optimization problem (2.32) to match the number and the type of the available
      sensor's measurements.
    Solve the optimization problem (2.32)
    }
end
```

5.3 Simulations and Experiments Data

The proposed localization scheme (Algorithm 5) was developed using ROS [118]. Here, the optimization problem (2.32) is formulated symbolically using CasADi: an open-source software tool for numerical optimization [9]. The optimization problem is then solved using the interior point method [54].

In order to validate the MHE localization scheme, several numerical and experimental datasets are used: 1) a simulated data of 3D motion of an aerial vehicle; 2) several sequences from EU long-term dataset for autonomous driving (see Section 4.4.1); and 3) experimental sequences generated using Summit-XL Steel omni-directional mobile robot [126], see Fig. 3.5. All the experiments and dataset runs are performed using a computer with Intel i7-8850H 6-core processor running at 2.60 GHz and a 16 GB RAM. The computer is running Ubuntu version 16.04 and ROS Kinetic.

5.3.1 Aerial Vehicle Simulation

A simulated aerial vehicle is used for validation of the proposed MHE localization scheme. The simulated vehicle is commanded to perform a spiral motion and is assumed to be equipped with a camera running a visual odometry (data is simulated), a GPS sensor, a barometer to measure the altitude of the vehicle, and 6-axis IMU. Each of the sensors is assumed to have additive Gaussian noise with zero mean; Table 5.1 shows the measurements of each of the sensors as well as their covariance matrices. The simulated motion is generated using the same omni-directional model stated in Eq. 5.2 with the constant velocities $v_x = 0.2$ m/s, $v_z = 0.3$ m/s, and $\omega_z = 0.1$ rad/s.

Table 5.1: The simulated sensors measurements and covariances for the aerial vehicle simulation

Sensor	**Measurements**	**Covariance**
Camera (VO)	$x, y, z, \alpha, \beta, \gamma$	$0.1\mathbf{I}_{6\times6}$
GPS	x, y	$0.1\mathbf{I}_{2\times2}$
IMU	$\omega_x, \omega_y, \omega_y, a_x, a_y, a_z$	$0.1\mathbf{I}_{6\times6}$
Barometer	z	0.1

5.3.2 EU Long-Term Dataset

The proposed MHE localization algorithm is then tested through fusing the noisy GPS, the LOAM and the IMU measurements (see chapter 4.4 for more information). The localization algorithm is validated using 9 sequences of the UTBM dataset with a total of about 45 km driving distance. The covariance of the LOAM measurement is estimated using the DCE algorithm proposed in Chapter 4 for both fusion schemes.

5.3.3 Data from Omni-directional Robot

Several experiments were conducted using the robot (see Section 3.3.2 for more information) to validate the proposed MHE localization algorithm while using the VICON motion capture system to generate the ground-truth data. The sensor fusion was performed using measurements from the IMU, wheel odometry, and 2D LiDAR odometry [28].

5.3.4 Evaluation Metrics

The performance of the proposed MHE localization algorithm is compared against a UKF-based localization scheme implemented in [98]. Both schemes are evaluated by, the mean and the maximum error percentages of the translation shown in Eq. 4.33 and Eq. 4.34, and the mean and maximum errors of the orientation as shown in Eq. 4.35 and Eq. 4.36.

5.4 Results and Discussion

In this section, the results of implementing Algorithm 5 are presented for the simulated aerial vehicle, UTBM dataset, and summit-XL Steel omni-directional robot.

5.4.1 Simulated Aerial Vehicle

As mentioned earlier, a spiral motion of an aerial vehicle was simulated for validating the MHE localization scheme. Fig. 5.4 (left) shows the 3D path taken by the simulated drone as well as the MHE and UKF estimation outputs. It can be observed that the MHE estimation is smoother and more accurate than that of the UKF.

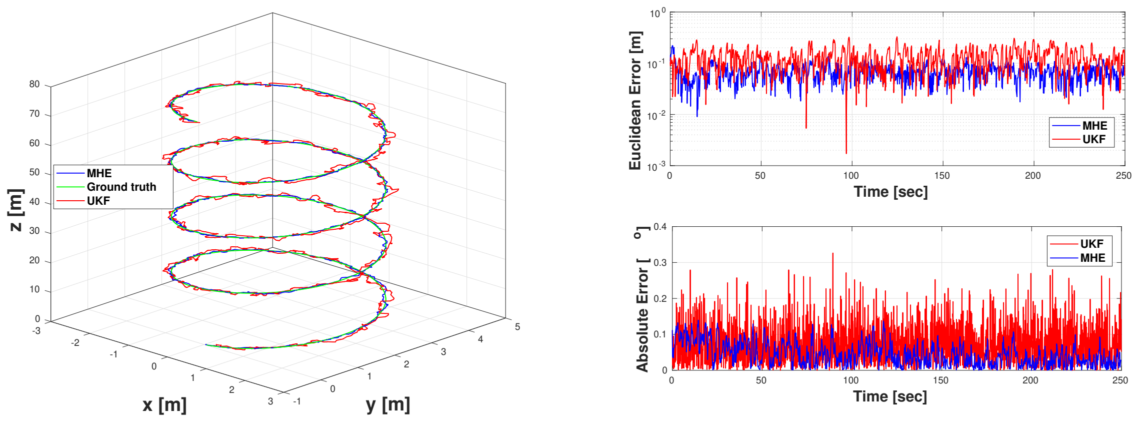


Figure 5.4: Left: The estimated trajectory by MHE and UKF as well as the ground truth of the simulated aerial vehicle spiral motion. Right: The translation and orientation errors in MHE and UKF for the simulated aerial vehicle spiral motion.

Fig. 5.4 (right) shows the error in the estimation of the MHE and UKF. The figure shows the superior performance of the MHE compared to UKF for both translation as well as orientation estimation. Table 5.2 shows the quantitative results of the aerial vehicle simulation. The table shows that the MHE estimation is almost twice as accurate as the UKF.

In order to further validate these results with real experimental data. The following two sections show the results of the proposed MHE fusion implementation using 2D sequences from EU Long-term dataset as well as the omni-directional Summit-XL steel robot. In order to show the feasibility of the algorithm for practical purposes, the computation time of the algorithm is also studied using one of the UTBM sequences.

Table 5.2: Mean and maximum translation and orientation errors of MHE ($N = 10$) and UKF for the simulated spiral path (total distance = 90 m).

Fusion Algorithm	TE_{mean}	TE_{max}	OE_{mean}	OE_{max}
MHE	**0.076**	**0.2457**	**0.02796**	**0.0888**
UKF	0.1257	0.3291	0.04579	0.20774

5.4.2 EU Long-Term Dataset

In Fig. 5.5, the MHE output as well as the measurements from the GPS and LOAM are presented. As can be seen in the figure, the MHE results show very accurate pose estimation as well as smooth trajectory despite the jumpy and discontinuous GPS measurements. As for LOAM measurements, in order to limit the amount of drift due to motion increments integration, the MHE or UKF output was used as a corrective feedback in order to correct for the drift error when fusing the odometry measurements (as was discussed in Section 4.3.2). This correction along with the Mahalanobis distance led to better fusion results through mitigating the effect of drift from the LiDAR odometry on the fusion output. Furthermore, throughout these sequences, the lateral velocity v_y was constrained to a small value $|v_y| < 0.1$ so that the prediction model of the MHE would be realistic for an Ackermann steering model. Such a constraint enhances the orientation estimation accuracy of the vehicle.

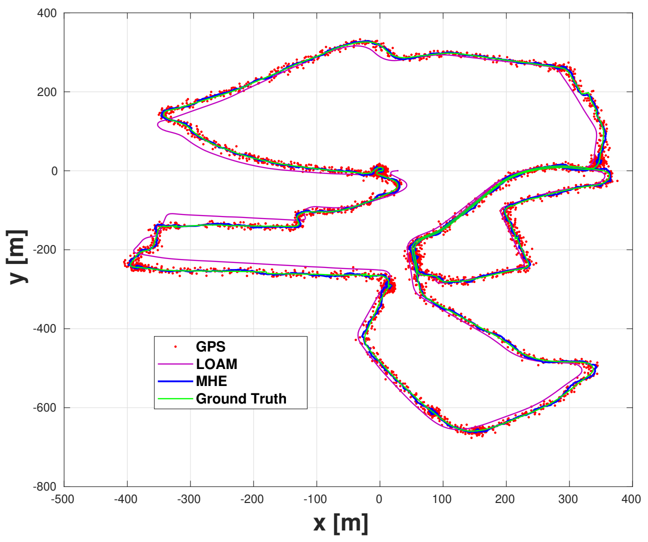


Figure 5.5: The estimated trajectory by MHE plotted with the GPS and the LOAM measurements (without correction) as well as the ground truth data for sequence 2018-05-02 (1) from EU long-term dataset.

Fig. 5.6 show the estimation output from the MHE and UKF as well as the translation and orientation errors along the path of the sequence 2018-05-02 (2) from EU long-term dataset. As can be seen in the Fig. 5.6 (left), the MHE output is smoother and more accurate when compared to the output of the UKF. Furthermore, as can be seen in Fig. 5.6

(right), the maximum translation and orientation errors in case of MHE are about 18 m and 12^o compared to a 35 m and 15^o in case of UKF.

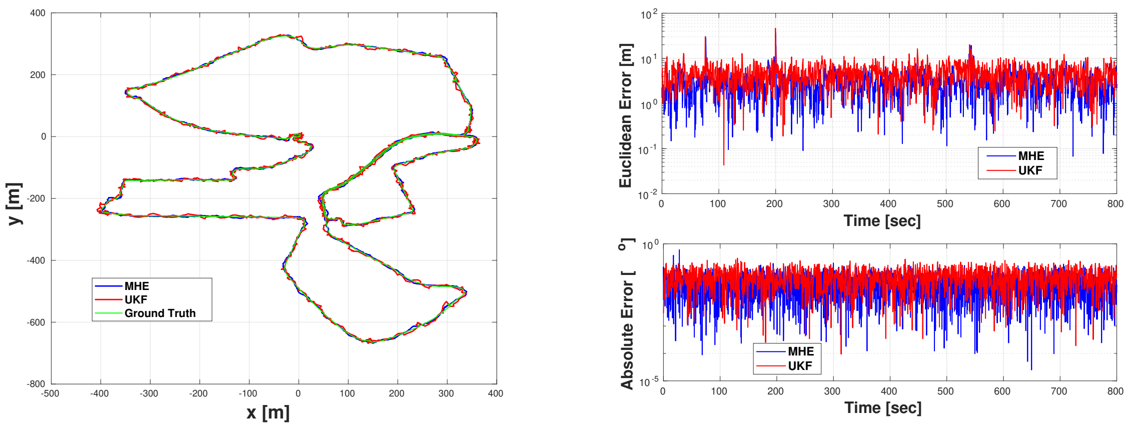


Figure 5.6: Left: The estimated trajectory by MHE and UKF as well as the ground truth for sequence 2018-05-02 (2) from EU long-term dataset. Right: Translation and orientation errors in MHE and UKF pose estimation for sequence 2018-05-02 (2) from EU long-term dataset.

Fig. 5.7 shows the MHE and UKF estimated paths for sequence 2019-01-31, which is slightly different and longer than the rest of the sequences. Notice that during this sequence, there are missing GPS measurements as can be seen in the figure. At this points, the MHE scheme managed to estimate the path correctly with minimal error while in case of the UKF, the output diverged until the GPS measurements became available again. This is in fact expected because the UKF algorithm depends on the latest measurements and prediction in order to estimate the pose while the MHE uses a window of old measurements for estimation ($N = 10$ in this case). This adds more robustness against missing measurements and allows for rejecting outliers without degrading the stability of the estimator. Furthermore, again in this sequence, the MHE shows much smoother estimation of the vehicle path compared with the UKF.

The overall quantitative results of the MHE fusion are shown in Table 5.3. The results show that the MHE fusion outperforms UKF fusion in every driving sequence of the EU Long-term dataset. The overall percentage error over the nine sequences is 0.05949% which is about 30% lower than the UKF estimation equivalent to 17 m Euclidean error in the nine sequences. This is also the case for orientation accuracy which is better by $0.0003^o/m$

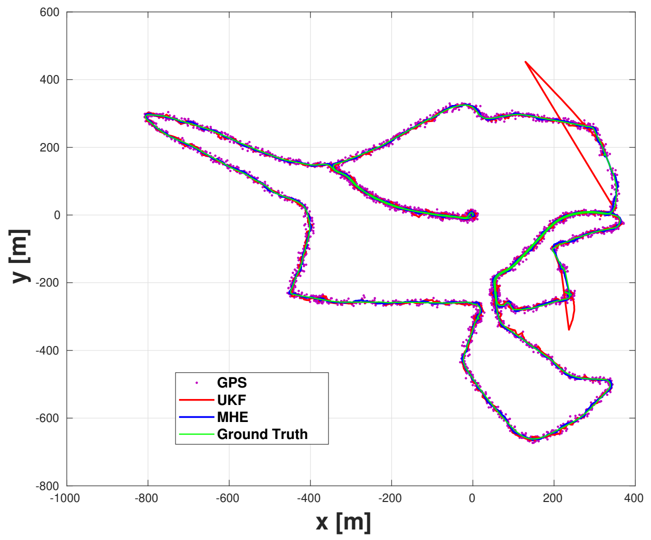


Figure 5.7: The estimated trajectory by MHE and UKF as well as the ground truth and GPS measurements for sequence 2019-01-31 from EU long-term dataset.

or about 14^o overall. This is a considerable enhancement in the performance of the pose estimator and the sensor fusion module for this practical experimental EU dataset.

Such enhancement in the pose estimation comes with the cost of increased computation time due to solving a constrained optimization problem every time-step. Table 5.4 shows the mean translation error for different estimation horizon sizes as well as the average computation time for each time-step. Notice that the estimation accuracy increases alongside the estimation horizon size. However, this in turn increases the computation cost of the algorithm. The results stated above in Table 5.3 were gathered while both algorithms (UKF and MHE) were running at a frequency of 5 Hz. The average computational time of UKF is 51 ms which is lower than the computational time of MHE even for a prediction horizon of $N = 2$. However, although the MHE fusion increases the computational time, the values indicated in Table 5.4 are still acceptable for practical purposes since, for example, for $N = 10$, the estimation algorithm can work with a frequency up to 6 Hz. Furthermore, for higher estimation frequencies, lower estimation horizons could be used to increase the speed of the fusion algorithm while maintaining better estimation accuracy compared to filtering techniques such as UKF.

Table 5.3: Mean and maximum translation and orientation errors of MHE ($N = 10$) and UKF for UTBM sequences.

Seq. Date	**MHE**				**UKF**				**Overall Distance**
	TE_{mean}	TE_{max}	OE_{mean}	OE_{max}	TE_{mean}	TE_{max}	OE_{mean}	OE_{max}	
2018-05-02 (1)	**0.0593**	**0.1579**	**0.00049**	**0.00253**	0.0738	0.1928	0.00076	0.00296	5076
2018-05-02 (2)	**0.0597**	**0.3546**	**0.0005**	**0.00225**	0.0721	0.4007	0.00079	0.00311	5071
2018-07-13	**0.0618**	**0.1897**	**0.0005**	**0.00203**	0.0622	0.2337	0.00078	0.00316	5028
2018-07-16	**0.0563**	**0.2428**	**0.00049**	**0.00242**	0.0893	0.2920	0.00085	0.00609	5010
2018-07-17	**0.0573**	**0.4965**	**0.00049**	**0.00224**	0.0910	0.5557	0.00079	0.00312	5063
2018-07-19	**0.0577**	**0.2081**	**0.0005**	**0.00241**	0.0905	0.3431	0.00078	0.00348	4980
2018-07-20	**0.0622**	**0.1939**	**0.0005**	**0.00224**	0.0897	0.2677	0.00083	0.00313	5001
2019-01-31	**0.0553**	**0.2149**	**0.00038**	**0.00184**	0.1343	7.4508	0.00069	0.00465	6443
2019-04-18	**0.0664**	**0.3356**	**0.0005**	**0.00276**	0.0885	0.3409	0.00078	0.00312	5060
Overall	**0.05949**	**0.2648**	**0.00048**	**0.00229**	0.0893	1.3102	0.00078	0.00367	46740

Table 5.4: Average translational error and computation time for different estimation horizons in MHE estimation in sequence 2018-08-02 (1) from EU long-term dataset.

Horizon N	TE_{mean}	OE_{mean}	**Mean Comp. Time (ms)**
2	0.1076	0.01496	54
3	0.09953	0.00790	58
4	0.07571	0.00119	73
5	0.06984	0.00078	89
10	0.05938	0.00049	148
20	0.06471	0.00059	325

5.4.3 Summit-XL Steel Omni-directional Robot

Using the Summit-XL Steel robot shown in Fig. 3.5, three sequences of data were generated for validating the MHE localization algorithm; the generated data are from executing 2 rectangular paths and one circular path.

In Fig. 5.8, the ground-truth taken by the VICON is plotted as well as each individual sensor measurement and the pose estimation using the proposed localization scheme. As can be seen, the MHE output is more accurate compared to both Encoder and LiDAR odometries. Furthermore, the output of the estimation is smooth enough, which makes it more suitable for navigation and control purposes.

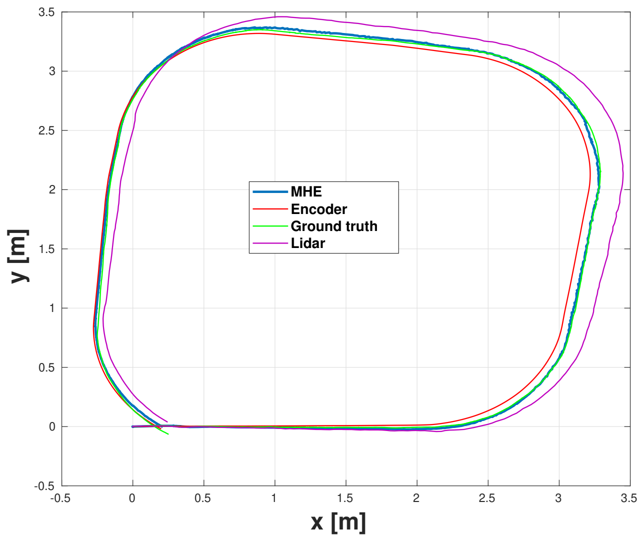


Figure 5.8: The estimated trajectory by MHE as well as the sensors measurement and the ground truth for rectangle sequence (2) executed by Summit-XL Steel omni-directional robot.

In Fig. 5.9 (left), another rectangular path is plotted along with the ground-truth data and the estimation output from MHE and UKF algorithms. The figure shows that the accuracy of the MHE algorithm is higher than that of the UKF. Furthermore, the estimation of the UKF shows oscillations in the estimated path by comparison to the estimated path by MHE which is much smoother.

In Fig. 5.9 (right), the translation and orientation errors for the rectangular path (1) are reported. As can be seen, the performance of MHE is better than that of UKF. Furthermore, it should be noted that although at some instances the estimation error in UKF seems to be better than that of the UKF, this can be attributed to the fact that the oscillations in the UKF were closer to the ground-truth. This may show that the results of the UKF at some instances are more accurate compared to MHE; however, such oscillations indicated lower robustness of the UKF output.

For further validation of the MHE algorithm, a circular path was executed using the robot. As can be seen in Fig. 5.10, the MHE estimation output is more accurate compared to that of different sensors odometries as well as the output of UKF fusion algorithm. Furthermore, in this scenario, the stability of the MHE algorithm is much more obvious while the output of UKF suffered from large amount of oscillations. The MHE algorithm,

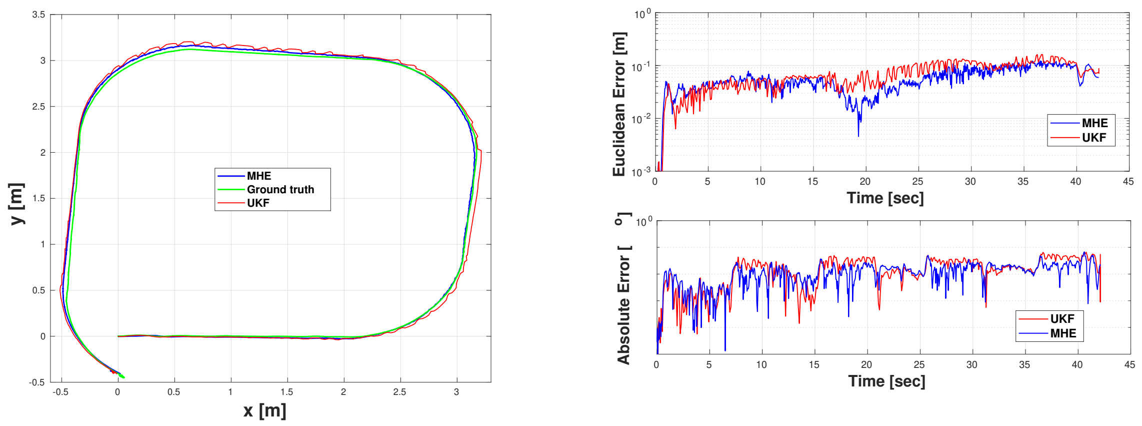


Figure 5.9: Left: The estimated trajectory by MHE and UKF as well as the ground truth for a rectangle (1) executed by Summit-XL Steel omni-directional robot. Right: Euclidean error (top) and absolution orientation error (bottom) in MHE and UKF pose estimation for rectangle sequence (1) Summit-XL Steel robots.

on the other hand, did not suffer from any considerable oscillations in its output.

Finally, Table 5.5 shows the quantitative results from Summit-XL Steel robot sequences. The translations and orientation errors from both MHE and UKF estimation output are reported for the three scenarios. It can be seen that MHE outperforms UKF in all cases and for all evaluation metrics.

Table 5.5: Mean and Maximum Translation Errors of MHE ($N = 10$) and UKF for Summit-XL Steel sequences.

Scenario Name	**MHE**				**UKF**				**Distance** [m]
	TE_{mean}	TE_{max}	OE_{mean}	OE_{max}	TE_{mean}	TE_{max}	OE_{mean}	OE_{max}	
Rectangle (1)	**0.4547**	**0.9722**	**0.06267**	**0.3041**	0.5769	1.2801	0.0930	0.2976	12.6
Rectangle (2)	**0.3152**	**0.7072**	**0.05891**	**0.2711**	0.3504	0.8376	0.09490	0.3134	12.5
Circle	**0.3446**	**1.3476**	**0.08496**	**0.8477**	0.4184	5.0123	0.1516	2.8466	6.5
Overall	**0.3768**	**0.9446**	**0.06577**	**0.4028**	0.4547	1.8727	0.1058	0.8282	31.6

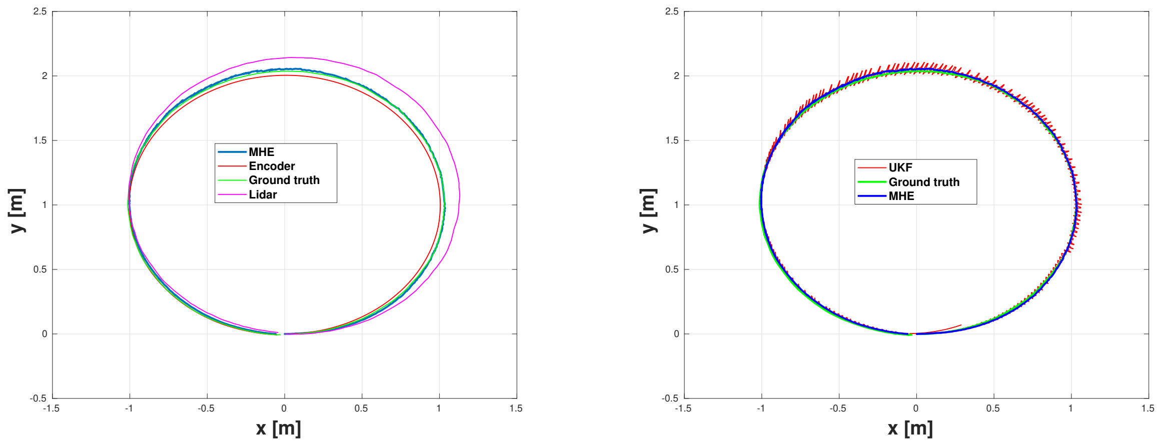


Figure 5.10: Left: The estimated trajectory by MHE as well as the sensors measurement and the ground truth for a circle sequence executed by Summit-XL Steel omni-directional robot. Right: Comparing MHE output to UKF output for the circular path.

5.5 Conclusion

In this chapter, a generic multi-sensor fusion scheme using MHE was proposed for the localization of autonomous vehicles. The proposed scheme considers several aspects to facilitate its practical deployment. These aspects include: sensors with different update rates, missed measurements and outlier rejection; different vehicle types; and real-time applicability. Moreover, this chapter provides an open-source implementation of the proposed localization scheme to aid its numerical/experimental validation by other practitioner in the autonomous systems community.

The proposed fusion scheme was tested using data generated numerically and experimentally, i.e autonomous driving sequences from EU Long-term dataset as well as experimental sequences using Summit-XL Steel omni-directional mobile robot. The MHE localization output was compared against that of the UKF. The MHE results showed superior accuracy in all cases as well as better stability and robustness, which make the proposed localization scheme more suitable for autonomous vehicles navigation and control.

Chapter 6

Mobile Manipulator Model Predictive Control[1]

6.1 Introduction

Chapter 3, 4 and 5 proposed different approaches to enhancing the localization accuracy of an autonomous platform. Accurate localization is very important for autonomous operation especially when it comes to the mobile platform navigating in unstructured environment. In this chapter, the autonomous navigation of a mobile robot is addressed through designing a point-stabilization controller of mobile manipulators end-effectors using Model Predictive Control (MPC).

Mobile manipulators combine the advantages of both wheeled robots and robotic arms; thus, they have an expandable workspace and operational versatility through perception, object manipulation, and mobility. Such robots can be used in material handling, wall painting, as well as inspection and repairs, see, e.g. [17] for a survey. Operating such systems requires safe navigation in possibly dynamic environments as well as precise object manipulation. In this chapter, we focus on stabilizing the end-effector of a 10-DOF mobile manipulator shown in Fig. 6.1. The figure shows the experimental platform available for this research. It is built for efficient autonomous object handling in warehousing applications by integrating the Summit-XL Steel mobile robot with meccanum wheels manufactured by Robotnik and the 7-DOF Barrett WAM robotic arm.

[1]This chapter was published as Mostafa Osman, Mohamed W. Mehrez, Shiyi Yang, Soo Jeon and William Malek, "End-Effector Stabilization of a 10-DOF Mobile Manipulator using Nonlinear Model Predictive Control," in *Proc. IFAC world congress*, Germany: Berlin, 2020. [112].

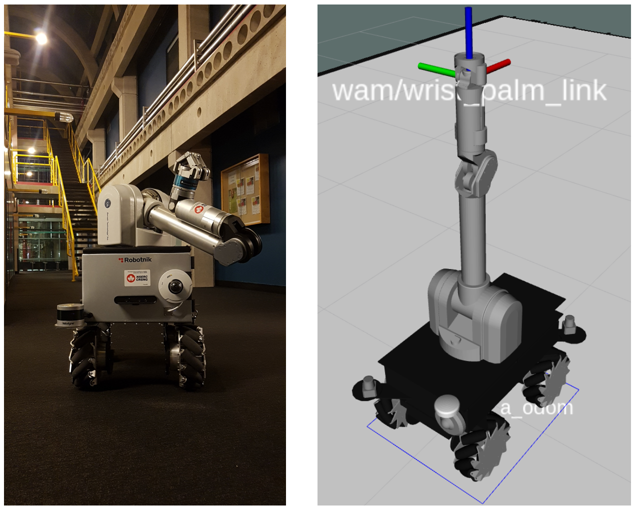


Figure 6.1: The synthesized mobile manipulator. Left: the real robot. Right: the simulated robot.

In this chapter, we use a Nonlinear Model Predictive Control (NMPC) scheme to stabilize the end-effector of a 10-DOF mobile manipulator; herein, we first formulate the task-space kinematic model, where the overall system rotations are expressed using the 3D special orthogonal group $SO(3)$ representation. Afterwards, this model is used for state prediction in the NMPC formulation, which considers state and control constraints as well as kinematic singularity constraints. The proposed NMPC controller is implemented using ROS [118] and the efficacy of the proposed controller is demonstrated through a series of real-time simulations using Gazebo dynamic simulator [68]. The results show highly-accurate end-effector positioning accuracy and smooth stabilization of the end-effector as well as computational cost, which meets the real-time applications requirements.

The remainder of this chapter is organized as follows: the model used to implement the NMPC is explained in Section 6.2 followed by the optimal control (OCP) problem formulation in Section 6.3. In Section 6.4, the simulation testbed used to validate the proposed controller is introduced and then the acquired results are shown in Section 6.5. Finally, in Section 6.6 conclusions are summarized.

6.2 Mobile Manipulator Modeling

In this section, we present the kinematic model of the synthesized mobile manipulator.

6.2.1 Kinematic Model of the holonomic Mobile Base

The mobile base of the considered mobile manipulator is a holonomic mobile robot with meccanum wheels. Holonomic mobile robots possess an extra degree of maneuverability when compared to the non-holonomic counterparts [138].

The discrete-time kinematic model of the mobile base is given by

$$\mathbf{x}_{k+1}^b = \mathbf{f}_{mr}(\mathbf{x}_k^b, \mathbf{u}_k) = \begin{bmatrix} x_k^b \\ y_k^b \\ \gamma_k \end{bmatrix} + \tau \underbrace{\begin{bmatrix} \cos\gamma_k & -\sin\gamma_k & 0 \\ \sin\gamma_k & \cos\gamma_k & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_b^I} \begin{bmatrix} u_{1,k} \\ u_{2,k} \\ u_{3,k} \end{bmatrix},$$

where $\mathbf{x}^b = [x, y, \gamma]^\top \in X_{mr} \subset \mathbb{R}^3$ is the pose of the mobile base in the inertial frame, $[x^b, y^b]$ are the two Cartesian planar coordinates and γ is the yaw angle of the mobile base. $\mathbf{u} = [u_1, u_2, u_3]^\top \in U_{mr} \subset \mathbb{R}^3$ is the robot input vector, $\mathbf{f}_{mr} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3$ is a nonlinear mapping, $R_b^I \in SO(3)$ is the z-axis rotation matrix which is the homogeneous transformation mapping from the robot frame to the inertial frame, and $\tau > 0$ is the sampling time.

The state constraint set X_{mr} is a compact set defined as

$$X_{mr} := [\underline{x}^b, \bar{x}^b] \times [\underline{y}^b, \bar{y}^b] \times [-\pi, \pi],$$

where $\underline{x}^b, \underline{y}^b, \bar{x}^b, \bar{y}^b$ are the lower and upper bounds of the Cartesian coordinates x^b and y^b, respectively.

The relation between the robot input vector $\mathbf{u}$ and wheel speeds $\mathcal{V} = (\omega_1, \omega_2, \omega_3, \omega_4)^\top$, for meccanum wheeled robots, can be stated as [89]

$$\mathcal{V} := \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \mathbf{H}\mathbf{u} = \frac{1}{\rho} \begin{bmatrix} 1 & -1 & -l-g \\ 1 & 1 & l+g \\ 1 & -1 & l+g \\ 1 & 1 & -l-g \end{bmatrix} \mathbf{u}, \tag{6.1}$$

where ρ is the mecanum wheel radius, l and g are half of the wheelbase and the trackwidth, respectively. Consequently, the control constraint set U_{mr} is a compact set defined as

$$U_{mr} := \{\mathbf{u} \in \mathbb{R}^3 | \ ||\mathbf{H}\mathbf{u}||_\infty \leq \omega_{max}\}, \tag{6.2}$$

where ω_{max} is the rated speed of the wheel motors.

6.2.2 Kinematic Model of the Robotic Arm

The robotic arm mounted on the aforementioned mobile base is a 7-DOF WAM arm by Barrett Technology [13]. The end-effector pose of the robotic arm is denoted by $\mathbf{x}^a := [\mathbf{p}^{a\top}, \theta^{a\top}]^\top$, where $\mathbf{p}^a \in \mathbb{R}^3$ is the end-effector position in the robotic arm base frame represented in the Cartesian coordinates and θ^a is the end-effector orientation. The end-effector orientation can be represented using several methods as discussed in [24]. Here, we use the $SO(3)$ group to avoid representation singularities. To do so, we define the mapping function $f : SO(3) \to \mathbf{F} \subset \mathbb{R}^9$, such that, for a rotation matrix $R \in SO(3)$,

$$f(R) = \begin{bmatrix}[R]_1^\top & [R]_2^\top & [R]_3^\top\end{bmatrix}^\top, \tag{6.3}$$

where $[R]_i$, $i \in \{1, 2, 3\}$, is the i-th column vector of the rotation matrix R. Thus, the orientation vector θ^a is defined as $\theta^a = f(R_E^b) \in \mathbf{F}$, and R_E^b is the rotation matrix of the end-effector relative to the base frame of the robotic arm.

Using such a representation, the kinematic model of the robotic arm can be described using the analytical Jacobian $\mathbf{J}_a$ of the forward kinematics transformation matrix $\mathcal{T} \in SE(3)$ derived using the DH-parameters of the robotic arm, (see [13] for the DH-parameters of the considered robotic arm), as

$$\mathbf{x}_{k+1}^a = \mathbf{f}_{ra}(\mathbf{x}_k^a, \mathbf{q}_k^a, \dot{\mathbf{q}}_k^a) = \mathbf{x}_k^a + \tau \mathbf{J}_a(\mathbf{q}_k^a)\dot{\mathbf{q}^{\mathbf{a}}}_k, \tag{6.4}$$

where $\mathbf{x}^a = [\mathbf{p}^{a\top}, \theta^{a\top}]^\top \in X_{ra} \subset \mathbb{R}^{12}$ is the state vector defined using θ^a from (6.3), $\mathbf{q}^a = [q_1, q_2, q_3, q_4, q_5, q_6, q_7]^\top \in Q \subset \mathbb{R}^7$ is the joint angles vector, $\dot{\mathbf{q}}^a \in \Omega \subset \mathbb{R}^7$ is the joint velocities vector, and, the analytical Jacobian $\mathbf{J}_a$ is given by $\mathbf{J}_a := \partial\mathcal{T}/\partial\mathbf{q}^a$.

The constraint sets for the end-effector X_{ra}, joint angles Q, and joint velocities Ω are defined by

$$\begin{aligned}
X_{ra} &:= [\underline{x}^a, \bar{x}^a] \times [\underline{y}^a, \bar{y}^a] \times [\underline{z}^a, \bar{z}^a] \times \mathbf{F},\\
Q &:= \{\mathbf{q}^a \in \mathbb{R}^7 | \underline{q}_i \le q_i \le \bar{q}_i, \forall i \in \{1, ..., 7\}\},\\
\Omega &:= \{\dot{\mathbf{q}}^a \in \mathbb{R}^7 |\ ||\dot{\mathbf{q}}^a||_\infty \le \dot{q}_{max}\},
\end{aligned}$$

where $\underline{q}_i$ and $\bar{q}_i$ denote the lower and upper limits of the joint angles, respectively. $||\dot{\mathbf{q}}^a||_\infty$ is the l_∞-norm defined as $||\dot{\mathbf{q}}^a||_\infty := \max_{i\in[1:7]} |\dot{q}_i^a|$.

In order to be able to keep track of the joint angles given the constraint set above, we extend system (6.4) to

$$\begin{bmatrix}\mathbf{x}_{k+1}^a\\ \mathbf{q}_{k+1}^a\end{bmatrix} = \begin{bmatrix}\mathbf{x}_k^a\\ \mathbf{q}_k^a\end{bmatrix} + \tau \begin{bmatrix}\mathbf{J}_a(\mathbf{q}_k^a)\\ I_{7\times7}\end{bmatrix} \dot{\mathbf{q}}_k^a, \tag{6.5}$$

where $[\mathbf{x}^{a\top}, \mathbf{q}^{a\top}]^\top \in \bar{X}_{ra} \subset \mathbb{R}^{19}$ is the concatenated state vector, and $\bar{X}_{ra}$ is the state constraint set for the new augmented model and is defined as $\bar{X}_{ra} := X_{ra} \times Q$.

6.2.3 Mobile Manipulator Kinematic Model

The model of the mobile manipulator can now be derived using the model of the mobile base and the robotic arm. First, we map the velocity components of the mobile base to the end-effector linear speeds in the inertial frame, i.e. we have

$$\dot{\mathbf{p}}^b = [R_b^I]_{2\times 2}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \underbrace{[R_b^I]_{2\times 2}\begin{bmatrix} -y_E^b \\ x_E^b \end{bmatrix}}_{\psi} u_3, \tag{6.6}$$

where $\dot{\mathbf{p}}^b = [\dot{x}^b, \dot{y}^b]$ is the linear velocity of the end-effector caused by the mobile base in the inertial frame, and $[R_b^I]_{2\times 2}$ is the upper left 2×2 sub-matrix of R_b^I shown in Eq. (6.1). Note that R_b^I is the orientation of the mobile robot, which can be determined through the localization feedback (e.g. Fig. 4.1. x_E^b, y_E^b are the position of the end-effector in the mobile base frame and can be determined from the forward kinematics transformation matrix $\mathcal{T}$.

Second, to calculate the angular velocity of the mobile robot in $SO(3)$, we need to compute the derivative of the rotation matrix R_b^I. The derivative of a rotation matrix R can be calculated as [24]

$$\dot{R} = S(\omega)R, \tag{6.7}$$

where $S(\omega)$ is the skew symmetric matrix form of the angular velocity vector $\omega = [\omega_x, \omega_y, \omega_z]^\top$, where ω_x, ω_y and ω_z are the three angular velocities around the principle axes x, y and z, respectively. Using the skew-symmetric matrix property $S(\mathbf{a})\mathbf{b} = -S(\mathbf{b})\mathbf{a}$, Eq. (6.7) can be written as

$$\dot{\theta}^b := \begin{bmatrix} [\dot{R}]_1 \\ [\dot{R}]_2 \\ [\dot{R}]_3 \end{bmatrix} = -\underbrace{\begin{bmatrix} S([R]_1) \\ S([R]_2) \\ S([R]_3) \end{bmatrix}}_{=:\Theta\in\mathbb{R}^{9\times 3}} \omega, \tag{6.8}$$

where $\dot{\theta}^b$ is the rate of change of the end-effector orientation due to the mobile base rotation. Moreover, $S([R]_i)$ is the i-th column vector of R in the skew symmetric form.

Since we do not consider any angular velocities other than w_z for the mobile base, the first two columns of Θ in (6.8) will be zeros. Consequently, $\dot{\theta}^b$ can be written as

$$\dot{\theta}^b := \begin{bmatrix} [\dot{R}_b^I]_1 \\ [\dot{R}_b^I]_2 \\ [\dot{R}_b^I]_3 \end{bmatrix} = -\underbrace{\begin{bmatrix} S([R_b^I]_1) \\ S([R_b^I]_2) \\ S([R_b^I]_3) \end{bmatrix}_3}_{=:\Theta_3\in\mathbb{R}^{9\times 1}} u_3, \tag{6.9}$$

where we exploit the fact that w_z for an omni-directional mobile robot is the control action u_3 shown in Eq. (6.1).

Using Eq. (6.6) and (6.9), in addition to the kinematic model in Eq. (6.5), the kinematic model of the whole mobile manipulator can be written as

$$\begin{aligned} \begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{q}^a_{k+1} \end{bmatrix} &= \begin{bmatrix} \mathbf{x}_k \\ \mathbf{q}^a_k \end{bmatrix} + \tau \mathbf{J}_{mm}(\mathbf{x}_k, \mathbf{q}^a_k) \begin{bmatrix} \mathbf{u}_k \\ \dot{\mathbf{q}}^a_k \end{bmatrix} \qquad (6.10) \\ &= \underbrace{\begin{bmatrix} \mathbf{x}_k \\ \mathbf{q}^a_k \end{bmatrix} + \tau \left[\begin{array}{cc|c} [R^I_b]_{2\times 2} & \begin{matrix} \psi \\ 0 \end{matrix} & \mathbf{J}_a(\mathbf{q}^a_k) \\ \mathbf{0}_{10\times 2} & -\Theta_3 & \\ \hline \multicolumn{2}{c|}{\mathbf{0}_{7\times 3}} & \mathbf{I}_{7\times 7} \end{array} \right] \begin{bmatrix} \mathbf{u}_k \\ \dot{\mathbf{q}}^a_k \end{bmatrix}}_{\mathbf{f}_{mm}(\mathbf{x}_k, \mathbf{q}^a_k, \mathbf{u}_k, \dot{\mathbf{q}}^a_k)}, \end{aligned}$$

where $[\mathbf{x}^\top, \mathbf{q}^{a\top}]^\top \in X \subset \mathbb{R}^{19}$ is the concatenated state vector of the mobile manipulator, $[\mathbf{u}, \dot{\mathbf{q}}^{a\top}]^\top \in U \subset \mathbb{R}^{10}$ is the concatenated control vector. Here, $\mathbf{x} := [\mathbf{p}^\top, \theta^\top]^\top$ is the end-effector pose vector in the inertial frame.

The model stated in (6.10) is the complete kinematic model of the considered 10-DOF mobile manipulator consisting of a 3-DOF holonomic mobile base and a 7-DOF robotic arm. The constraints over the developed kinematic model can now be defined as

$$\begin{aligned} X &:= [\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}] \times [\underline{z}, \bar{z}] \times \mathbf{F} \times Q, \text{ and} \qquad (6.11) \\ U &:= U_{mr} \times \Omega. \end{aligned}$$

Finally, the end-effector pose feedback can be determined through using a localization scheme such as one of the visual odometries utilized in Chapter 3 or through the use of the multi-sensor fusion localization scheme proposed in Chapter 5, as well as through using the forward kinematic equations of the robotic arm $\mathcal{T}$.

6.3 Nonlinear Model Predictive Control

In this section, we formulate an NMPC scheme for the end-effector pose stabilization of the mobile manipulator in (6.10). To this end, we define

$$\begin{aligned} \mathcal{U}_{N_c} &:= \left(\begin{bmatrix} \mathbf{u}_k \\ \dot{\mathbf{q}}^a_k \end{bmatrix}, \begin{bmatrix} \mathbf{u}_{k+1} \\ \dot{\mathbf{q}}^a_{k+1} \end{bmatrix}, \dots, \begin{bmatrix} \mathbf{u}_{k+N_c-1} \\ \dot{\mathbf{q}}^a_{k+N_c-1} \end{bmatrix} \right) \text{ and} \\ \mathcal{X}_{N_c} &:= (\mathbf{x}_k, \mathbf{x}_{k+1}, \dots, \mathbf{x}_{k+N_c}) \end{aligned}$$

as the sequences of controls and states over the prediction horizon $N_c \in \mathbb{N}$, respectively. As standard in NMPC, these sequences are used to form the quadratic cost function

$$J_{MPC}(\mathcal{U}_{N_c}, \mathcal{X}_{N_c}) = \underbrace{||\mathcal{E}^{\mathbf{p}}_{N_c}||^2_{\mathbf{S^p}} + ||\mathcal{E}^{\theta}_{N_c}||^2_{\mathbf{S}^\theta}}_{J_f} \\ + \sum_{i=k}^{k+N_c-1} ||\mathcal{E}^{\mathbf{p}}_i||^2_{\mathbf{Q^p}} + ||\mathcal{E}^{\theta}_i||^2_{\mathbf{Q}^\theta} + \left\| \begin{matrix} \mathbf{u}_i \\ \dot{\mathbf{q}}^a_i \end{matrix} \right\|^2_{\mathbf{R}}, \tag{6.12}$$

where $\mathbf{S^p} \in \mathbb{R}^{3\times3} \succ 0, \mathbf{S}^\theta \in \mathbb{R}^{9\times9} \succ 0, \mathbf{Q^p} \in \mathbb{R}^{3\times3} \succ 0, \mathbf{Q}^\theta \in \mathbb{R}^{9\times9} \succ 0$ and $\mathbf{R} \in \mathbb{R}^{10\times10} \succ 0$ are the weighting matrices of the quadratic cost function. Define J_f as the terminal cost, $\mathcal{E}^{\mathbf{p}} \in \mathbb{R}^3$ as the translational error of the end-effector pose $\mathcal{E}^{\mathbf{p}} := \mathbf{p} - \mathbf{p^r}$, where $\mathbf{p}^r$ is the reference position, and $\mathcal{E}^\theta \in \mathbb{R}^9$ as the orientation error of the end-effector pose

$$\mathcal{E}^\theta := \begin{bmatrix} [I_{3\times3}]_1 \\ [I_{3\times3}]_2 \\ [I_{3\times3}]_3 \end{bmatrix} - \begin{bmatrix} [(R^I_E)^\top R_r]_1 \\ [(R^I_E)^\top R_r]_2 \\ [(R^I_E)^\top R_r]_3 \end{bmatrix}, \tag{6.13}$$

where R_r is the desired orientation, and R^I_E is the orientation of the end-effector in the inertial frame calculated as $R^I_E = R^I_b R^b_E$. R^I_b is determined using a localization algorithm of the mobile robot and R^b_E is the rotation matrix from the mobile robot to the end-effector calculated from the forward kinematics of the robotic arm, i.e. $\mathcal{T}$.

Using the cost function in Eq. (6.12), the NMPC Optimal Control Problem (OCP) can be formulated as:

$$(\mathcal{U}^*_N, \mathcal{X}^*_N) = \underset{\mathcal{U}_N \in U, \mathcal{X}_N \in X}{\arg\min} \; J_{MPC}(\mathcal{U}_N, \mathcal{X}_N) \tag{6.14a}$$

$$\text{subject to} \quad \begin{bmatrix} \mathbf{x}_{k+1} & \mathbf{q}^a_{k+1} \end{bmatrix}^\top - \mathbf{f}_{mm}(\mathbf{x}_k, \mathbf{q}^a_k, \mathbf{u}_k, \dot{\mathbf{q}}^a_k) = 0, \tag{6.14b}$$

$$\mathcal{X}_N \in X \subseteq \mathbb{R}^{19}, \tag{6.14c}$$

$$\mathcal{U}_N \in U \subseteq \mathbb{R}^{10}, \tag{6.14d}$$

$$|\det(\mathbf{J}_a \mathbf{J}_a^T)| > \epsilon_s \tag{6.14e}$$

where ϵ_s is a small number acting as a threshold to avoid singular configurations of the robotic arm which can be evaluation using inequality (6.14e) in the NMPC formulation.

OCP (6.14) is converted to a Nonlinear Programming Problem (NLP) using the direct multiple-shooting method [6]. Here, both the control sequence $\mathcal{U}_N$ as well as the state sequence $\mathcal{X}_N$ are considered as decision variables in the optimization problem. Moreover, the system model is considered as an optimization constraint formulated by Eq. (6.14b).

Multiple-shooting discretization technique provides a more computationally efficient solution to the OCP (6.14) when compared with other discretization techniques, e.g. single-shooting, see [6] for more details. Finally, state and control constraints are considered by means of Eq. (6.14c) and (6.14d). Note that the inequality constraint (6.14e) is added to avoid singular configurations when the system is in operation under the NMPC control scheme. This is accomplished through ensuring that the pseudo-inverse of the robot arm Jacobian matrix is always invertible and, thus, singular configurations are avoided.

The feedback control law can now be stated as

$$\begin{bmatrix}\mathbf{u}_k^* & \dot{\mathbf{q}}_k^{a*}\end{bmatrix}^\top := \mathcal{U}_N^*(0),$$

i.e. the feedback control is the first element in the optimal control sequence $\mathcal{U}_N^*$. Moreover, the resulting feedback system can be stated as

$$\begin{bmatrix}\mathbf{x}_{k+1} & \mathbf{q}_{k+1}^a\end{bmatrix}^\top = \mathbf{f}_{mm}(\mathbf{x}_k, \mathbf{q}_k^a, \mathbf{u}_k^*, \dot{\mathbf{q}}_k^{a*}).$$

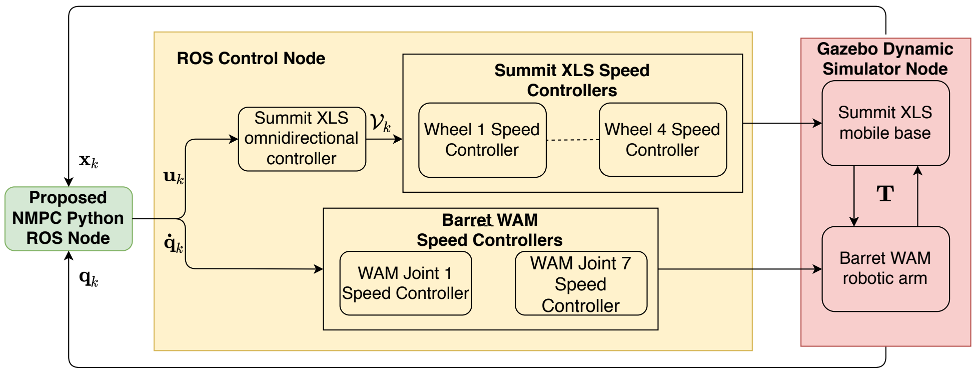
6.4 Simulation Environment and Test Scenarios

The simulation testbed of the considered mobile manipulator is created by synthesizing the "urdf" models of both the Barret WAM arm and the Summit XL Steel mobile robot in a ROS/Gazebo simulation environment[2]. Here, the transformation and the constraints between the two models are defined based on the actual physical system, see Fig. 6.1. Moreover, the proposed NMPC controller is programmed in python and integrated with the simulation environment via a ROS-node. Here, OCP (6.14) is formulated symbolically using the numerical optimization software tool CasADi [9]. Additionally, OCP (6.14) is solved using the interior-point optimization method via the open source solver IPOPT [157] (The library and optimizer used in Chapter 5). The overall block-diagram of the ROS/Gazebo real-time simulation environment is in Fig. 6.2.

The scenarios shown in Table 6.1 were used to validate the proposed controller. Initial and set-point references shown are presented using the Z-Y-X Euler angles to simplify the presentation. In all scenarios, the sampling time is $\tau = 0.15$ sec, the prediction horizon is $N_c = 5$, and the weighting matrices are $\mathbf{S}^\mathbf{p}, \mathbf{Q}^\mathbf{p} = 2\mathbf{I}_{3\times3}$, $\mathbf{S}^\theta, \mathbf{Q}^\theta = 15\mathbf{I}_{9\times9}$ and $\mathbf{R} = \mathbf{I}_{10\times10}$. Furthermore, the constraints set X defined in (6.11) is chosen as

$$X = [-3, 3] \times [-3, 3] \times [0.4, 1.43] \times \mathbf{F} \times Q,$$

[2]urdf: universal robotic description format.



ROS Control Node
Summit XLS Speed Controllers
Summit XLS omnidirectional controller
$\mathcal{V}_k$
Wheel 1 Speed Controller
Wheel 4 Speed Controller
Barret WAM Speed Controllers
WAM Joint 1 Speed Controller
WAM Joint 7 Speed Controller
Proposed NMPC Python ROS Node
$\mathbf{x}_k$
$\mathbf{u}_k$
$\dot{\mathbf{q}}_k$
$\mathbf{q}_k$
Gazebo Dynamic Simulator Node
Summit XLS mobile base
$\mathbf{T}$
Barret WAM robotic arm

Figure 6.2: Block-diagram of ROS/Gazebo dynamic simulation environment used to validate the proposed controller.

where the arm joint angles limits set Q is given by

$$Q = \left\{ \begin{bmatrix} -2,6 \\ -1.985 \\ -2.8 \\ -0.9 \\ -4.55 \\ -1.5707 \\ -3.0 \end{bmatrix} \leq \mathbf{q} \leq \begin{bmatrix} 2.6 \\ 1.985 \\ 2.8 \\ \pi \\ 1.25 \\ \pi/2 \\ 3.0 \end{bmatrix} \right\}.$$

Here, the joint angle limits are adapted from the arm specifications [13]. Finally, the limit of the joints speeds is chosen as $\dot{q}_{max} = 0.5$ rad/sec and the angular speed limit of the mobile robot wheels is chosen as $\omega_{max} = 0.6$ rad/sec.

6.5 Results and Discussion

In this section, we show the closed-loop results of the mobile manipulator under the NMPC controller for all the scenarios stated in Table 6.1. Fig. 6.3 shows the trajectory of the end-effector through each case and the actual real-time simulation is recorded and shown in the video in the link[3]. As shown in the figure, the controller successfully stabilized the end-effector of the mobile manipulator to the desired position.

[3]https://youtu.be/BplnroBEmGo

Table 6.1: Real-time simulation scenarios

Scenario	Initial States	Reference Pose
S1	$[2, 0, 1.42, 0, 0, 0]^\top$	$[0, 0, 0.5, \pi, 0, 0]^\top$
S2	$[2, 2, 1.42, 0, 0, 0]^\top$	$[0, 0, 0.5, \pi/2, 0, 0]^\top$
S3	$[0, 2, 1.42, 0, 0, 0]^\top$	$[0, 0, 0.5, 0, \pi/2, 0]^\top$
S4	$[-2, 2, 1.42, 0, 0, 0]^\top$	$[0, 0, 0.5, 0, \pi, 0]^\top$
S5	$[-2, 0, 1.42, 0, 0, 0]^\top$	$[0, 0, 0.5, 0, 0, \pi/2]^\top$
S6	$[-2, -2, 1.42, 0, 0, 0]^\top$	$[0, 0, 0.5, 0, 0, \pi]^\top$
S7	$[0, -2, 1.42, 0, 0, 0]^\top$	$[0, 0, 0.5, 0, 0, 0]^\top$
S8	$[2, -2, 1.42, 0, 0, 0]^\top$	$[0, 0, 0.5, \pi, 0, 0]^\top$

The performance of the NMPC controller is evaluated by calculating both position and orientation errors of the end-effector with respect to the reference pose. The positional error is measured by the Euclidean distance between the end-effector position and the reference position, while the orientation error is measured by the evaluation metric

$$E^\theta = 3 - \mathrm{Tr}(R_E^I R_r^\top), \tag{6.15}$$

where the trace of the error rotation matrix is used. In essence, Eq. (6.15) indicates that the orientation error E^θ converges to zero as the the end-effector orientation R_E^I converges to the reference orientation R_r.

As shown in Fig. 6.4(left), the positional error converges to zero for all scenarios considered. Note that in scenarios 6 and 7, the settling time of the controller is relatively larger than that for the other scenarios due to the joints limits. In these two cases, the mobile manipulator had to take a longer maneuver to reach the required position and orientation without violating any joint limits. In Fig. 6.4(right), the orientation error of the end-effector, i.e. E^θ, is shown. The error is calculated using the evaluation metric stated in (6.15). As can be seen in the figure, the controller managed to quickly achieve all the required orientations.

All the real-time simulations were executed using an Intel Core i7 CPU with 2.10 GHz processor. The average computation time of the OCP (6.14) throughout all the simulation scenarios was 61 ms with a maximum computation time of 130 ms and a standard deviation of 23.9 ms. Considering that the sampling time used is $\tau = 150$ ms, the computational

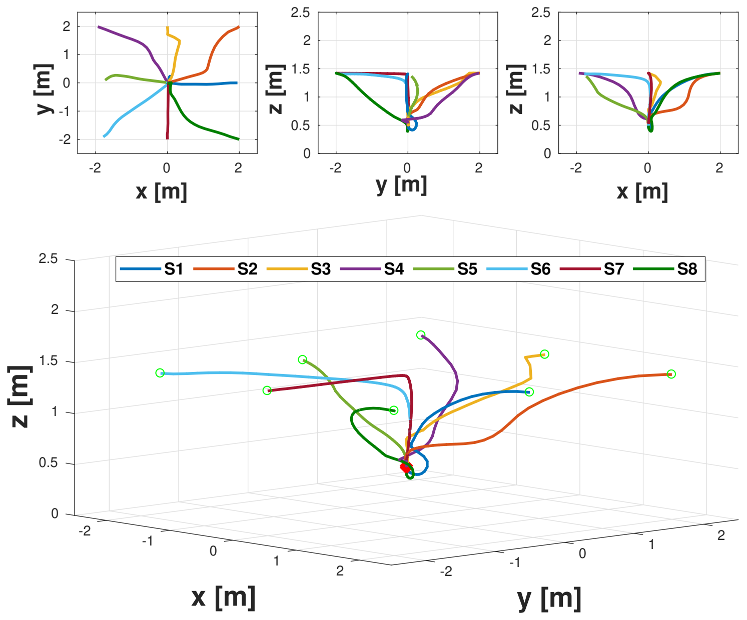


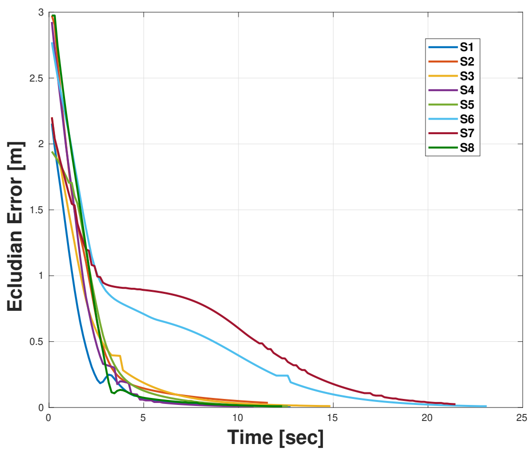
Figure 6.3: The position trajectories of the end-effector for all the scenarios. Bottom: 3D visualization of the trajectories taken by th end-effector. Top: The three projected views of the end-effector trajectories.

results suggest that the proposed NMPC algorithm meets the real-time implementation requirements while simultaneously generating feasible solutions.

6.6 Conclusion

In this chapter, we proposed an NMPC controller for end-effector stabilization of a 10-DOF mobile manipulator. We used the kinematic models for both the mobile base and the robot arm to realize a task-space control for the end-effector of the mobile manipulator. Required constraints include the joint limits and the manipulator singularity. Using the developed model along with NMPC, the stabilization of the end-effector was achieved without the need of any inverse kinematics solvers. Therefore, the proposed controller is a stand-alone high level controller, which only requires the localization feedback of the mobile base and the joint positions feedback of the robotic arm on the mobile manipulator.

The proposed controller was validated through real-time dynamic simulation scenarios. The results showed efficacy and efficiency of the proposed NMPC controller. Throughout all



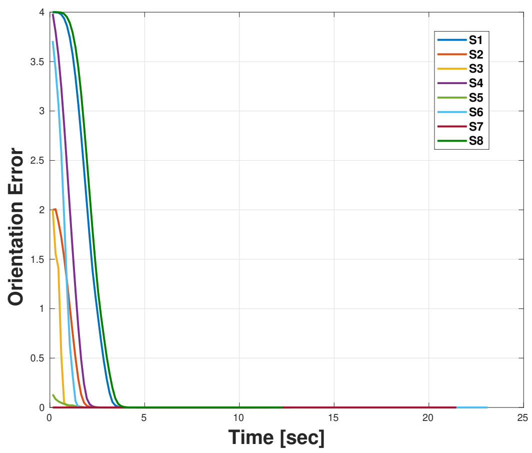


Figure 6.4: The position (left) and orientation (right) error of the end-effector for all scenarios.

the scenarios, the controller managed to smoothly stabilize the end-effector to the required pose.

Chapter 7

Conclusion and Future Work

7.1 Introduction

The research endeavors and conclusions are summarized in this chapter. Section 7.2 summarizes the thesis contributions and some concluding remarks. Moreover, Section 7.3 describes the future directions and possible development of the presented works.

7.2 Final Conclusion

In this thesis, several problems related to accurate localization of autonomous platforms were presented. Namely,

- visual odometry accuracy and robustness,
- uncertainty estimation in odometries, and
- accurate multi-sensor fusion-based localization.

Furthermore, a stabilization controller of a mobile manipulator based on MPC was introduced.

Throughout each chapter of this thesis, one of the objectives stated in Section 1.3 was addressed.

First, in Chapter 3, an image processing pipeline was introduced to enhance the accuracy and robustness of the VO algorithm. The proposed algorithm consisted of three stages, namely, CLAHE, SSC, and AOR. Each stage addresses one of the problems, which might be a cause of error in the pose estimation.

As was already discussed in Chapter 3, the three proposed stages played an integral part in enhancing the accuracy of the pose estimation in the VO algorithms used in the case studies. Furthermore, the three stages were a desirable combination to overcome the drawbacks of each individual stage, while utilizing their advantages.

The proposed pipeline is intended to be generic and modular, i.e. can be embedded in any feature-based algorithm in order to enhance its performance. The proposed pipeline was validated through using sequences from KITTI and TUM datasets as well as experimental sequences using a Summit XL Steel robot. Moreover, for each dataset, one type of VO was used for validation, namely stereo, RGB-D and monocular. The quantitative and qualitative results show that the proposed pipeline indeed serves as a desirable combination to enhance the VO accuracy and robustness, with the cost of slightly increasing the computational cost.

Second, in Chapter 4, the Drift Covariance Estimation (DCE) algorithm was introduced. DCE algorithm estimates the covariance of odometries (which suffer from accumulation of error due to integration) through the use of another sensor which does not suffer from drift. The DCE algorithm overcomes the challenges of quantifying the covariance by the presence of ground-truth or through hard-tuning and also takes into consideration the fact that the covariance of such odometries is dynamic and changes during operation.

The proposed DCE algorithm was tested using several experimental sequences from EU Long-term dataset. The localization output through using the DCE was then compared to the localization output using different constant covariances. All the results confirmed the fact that the use of the DCE resulted in a better performance compared with the use of constant covariance. Furthermore, the results showed that integrating an odometry with the DCE algorithm in a multi-sensor fusion scheme, causes an enhancement of the pose estimation compared to just using the odometry for pose estimation.

Third, in Chapter 5, in order to achieve better pose estimation through multi-sensor fusion-based localization, a generic multi-sensor fusion scheme using MHE was proposed for the localization of autonomous platforms. The proposed scheme considers several aspects for its practical deployment. These aspects include: sensors with different update rates, missed measurements and outlier rejection; different vehicle types; and real-time applicability.

The proposed fusion scheme was tested using numerically simulated data of an aerial vehicle, as well as experimental sequences from EU Long-term dataset and Summit XL

Steel omni-directional robot. The MHE localization output was compared against that of a UKF. The results showed that the MHE outperformed the UKF in terms of accuracy and robustness with the cost of increased computational cost.

Finally, in Chapter 6, an NMPC controller for end-effector stabilization of a 10-DOF mobile manipulator was introduced. After developing the overall kinematic model of the mobile manipulator using the $SO(3)$ group, and taking into consideration all the required constraints, the OCP was formulated and solved using the open-source library called CasADi and the open source optimizer IPOPT.

The controller was validated through real-time dynamic simulation scenarios. the results showed efficacy and efficiency of the proposed NMPC controller. Throughout all the designed scenarios, the controller managed to smoothly stabilize the end-effector to the required pose.

7.3 Future Work

The work proposed in this thesis can be further enhanced or integrated together to achieve better autonomous performance. Some of the directions that can be taken for further developing the work proposed in this thesis are:

- The work proposed in this thesis can be integrated to achieve better autonomous operation as shown in Fig. 7.1. As shown in the figure, a visual odometry algorithm augmented with the proposed pipeline in Chapter 3 can be used to achieve lower drift. Furthermore, using the DCE algorithm proposed in Chapter 4, the covariance of each of the odometries used can be estimated, which can then be used for accurate multi-sensor fusion by the MHE scheme proposed in Chapter 5. Finally, the localization output from the MHE can used as the feedback for the controller proposed in Chapter 6. The author aims to validate the efficacy of the whole system through real-world experiments.
- The proposed pipeline is planned to be integrated to visual SLAM algorithms and the effect of the pipeline will be studied. Several additional filtration steps will also be investigated for further enhancing the performance of VO algorithms. Meanwhile, the computational cost of the algorithm is expected to be reduced through the use of GPUs and parallel computing techniques.
- While the proposed MHE localization scheme meet the computational requirements up to a certain levels, further reduction of the computational cost could be achieved

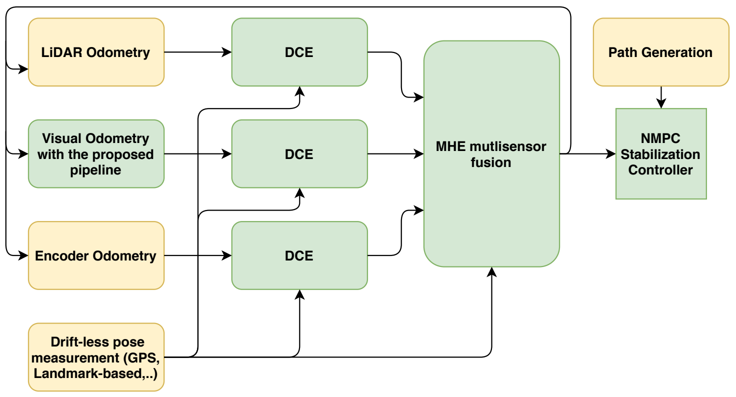

LiDAR Odometry
DCE
Visual Odometry with the proposed pipeline
DCE
Encoder Odometry
DCE
Drift-less pose measurement (GPS, Landmark-based,..)
MHE mutlisensor fusion
Path Generation
NMPC Stabilization Controller

Figure 7.1: An example for integrating the work for better autonomous operation where the green blocks show the work proposed in this thesis, while the yellow blocks were not addressed in this thesis.

through investigating code-generation techniques to reduce the optimization problem solution time. Moreover, different optimization libraries such as Google Ceres [29] will be tested for better optimization speed.

- The DCE algorithm proposed in Chapter 4 can be further extended to estimate the covariance of odometries through the use of the multi-sensor fusion output instead of an external sensor. In this case, the algorithm can be used, even for systems which only relies on different odometries for localization without any drift-free measurements.

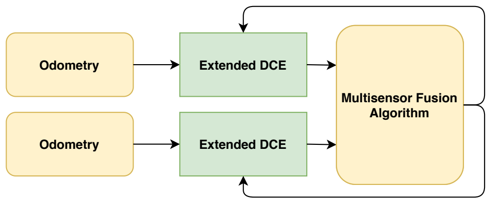


Figure 7.2: Covariance estimation using an extended version of the DCE algorithm through using the multi-sensor fusion algorithm output instead of using another drift-less sensor.

- Finally, real experiments will be executed using the proposed NMPC controller in Chapter 6 to further validate the controller in real-life scenarios. Additionally, the integration of the controller with path planning algorithms will be investigated. Finally, the path following problem for the same 10-DOF mobile manipulator will be studied.

List of Publications

Journal Papers

- **Mostafa Osman**, Ahmed Hussein, Abdulla Al-Kaff, Fernando Garcia, and Dongpu Cao, ”A Novel Online Approach for Drift Covariance Estimation of Odometries Used in Intelligent Vehicle Localization,” *Sensors*, 19.23 (2019): 5178 [111].
- Mohamed W. Mehrez, Karl Worthmann, Joseph P.V. Cenerini, **Mostafa Osman**, William Melek, and Soo Jeon, ”Model Predictive Control without terminal constraints or costs for holonomic mobile robots,” *Robotics and Autonomous Systems*, 127 (2020): 103468 [95].
- Mohamed Sabry, **Mostafa Osman**, Ahmed Hussein, Mohamed W. Mehrez, Soo Jeon, and William Melek, ”A Generic Image Processing Pipeline for Enhancing Accuracy and Robustness of Visual Odometry,” *IEEE Trans. on Intelligent Transportation Systems* [Still under submission].
- **Mostafa Osman**, Mohamed W. Mehrez, Mohamed A. Daoud, Ahmed Hussein, Soo Jeon, and William Melek, ”A Generic Multisensor Fusion Scheme for Localization of Autonomous Platforms Using Moving Horizon Estimation,” *Information Fusion*, Elsevier [Still under submission].

Conference Papers

- **Mostafa Osman**, Ricardo Alonso, Ahmed Hammam, Francisco Miguel Moreno, Abdulla Al-Kaff, and Ahmed Hussein, ”Multisensor Fusion Localization using Extended H_∞ Filter using Pre-filtered Sensors Measurements,” in: *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2019 [108].

- **Mostafa Osman**, Ahmed Hussein, and Abdulla Al-Kaff, ”Intelligent Vehicles Localization Approaches between Estimation and Information: A Review,” in: *Proc. IEEE Int. Conf. of Vehicular Electronics and Safety (ICVES)*, 2019 [109].
- Mohamed A. Daoud, **Mostafa Osman**, Mohamed W. Mehrez, and William W. Melek, ”Path-following and Adjustable Driving Behavior of Autonomous Vehicles using Dual-Objective Nonlinear MPC,” in: *Proc. IEEE Int. Conf. of Vehicular Electronics and Safety (ICVES)*, 2019 [31].
- **Mostafa Osman**, Mohamed W. Mehrez, Shiyi Yang, Soo Jeon, and William W. Melek, ”End-Effector Stabilization of a 10-DOF Mobile Manipulator using Nonlinear Model Predictive Control,” in *Proc. 21st IFAC World Congress*, 2020 [112].

References

[1] Reza Abbaspour. Design and implementation of multi-sensor based autonomous minesweeping robot. In *Proc. Int. Congress on Ultra Modern Telecommunications and Control Systems*, pages 443–449, 2010.

[2] Markus Achtelik. vicon bridge. *GitHub Repository*, [Online, Last Accessed: 31 May 2020].

[3] Shahrokh Akhlaghi, Ning Zhou, and Zhenyu Huang. Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation. *arXiv preprint arXiv:1702.00884*, pages 1–5, 2017.

[4] Abdulla Al-Kaff, Ricardo Alonso, Mostafa Osman, and Ahmed Hussein. Skyonyx: Autonomous uav research platform for air transportation system (atsys). In *Proc. 21st Int. Conf. on Intelligent Transportation Systems (ITSC)*, pages 3550–3555. IEEE, 2018.

[5] Mohamed Aladem and Samir A Rawashdeh. Lightweight visual odometry for autonomous mobile robots. *Sensors*, 18(9):2837, 2018.

[6] J. Albersmeyer and M. Diehl. The lifted newton method and its application in optimization. *SIAM J. on Optimization*, 20(3):1655–1684, 2010.

[7] Hatem Alismail, Michael Kaess, Brett Browning, and Simon Lucey. Direct visual odometry in low light using binary descriptors. *IEEE Robotics and Automation Letters*, 2(2):444–451, 2016.

[8] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[9] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 2018.

[10] Astra. Astra RGB-D Camera. https://orbbec3d.com/product-astra-pro/.

[11] Giovanni Buizza Avanzini, Andrea Maria Zanchettin, and Paolo Rocco. Reactive constrained model predictive control for redundant mobile manipulators. In *Intelligent Autonomous Systems 13*, pages 1301–1314. Springer, 2016.

[12] Oleksandr Bailo, Francois Rameau, Kyungdon Joo, Jinsun Park, Oleksandr Bogdan, and In So Kweon. Efficient adaptive non-maximal suppression algorithms for homogeneous spatial keypoint distribution. *Pattern Recognition Letters*, 106:53–60, 2018.

[13] Barrett. WAM Arm Specifications. https://advanced.barrett.com/wam-arm-1, 2018.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proc. European Conf. on Computer Vision*, pages 404–417. Springer, 2006.

[15] ERTUĞRUL BAYRAKTAR and Pinar Boyraz. Analysis of feature detector and descriptor combinations with a localization experiment for various performance metrics. *Turkish J. of Electrical Engineering and Computer Science*, 25(3):2444–2454, 2017.

[16] Johann Borenstein and Liqiang Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Trans. on Robotics and Automation*, 12(6):869–880, 1996.

[17] Roger Bostelman, Tsai Hong, and Jeremy Marvel. Survey of research for performance measurement of mobile manipulators. *J. of Research of the National Institute of Standards and Technology*, 121:342–366, 2016.

[18] Mattia Brambilla, Gloria Soatti, and Monica Nicoli. Precise vehicle positioning by cooperative feature association and tracking in vehicular networks. In *Proc. IEEE Statistical Signal Processing Workshop (SSP)*, pages 648–652, 2018.

[19] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 510–517, 2005.

[20] Martin Buczko and Volker Willert. How to distinguish inliers from outliers in visual odometry for high-speed automotive applications. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 478–483, 2016.

[21] Vladimir Bychkovskiy, Seapahn Megerian, Deborah Estrin, and Miodrag Potkonjak. A collaborative approach to in-place sensor calibration. In *Proc. Information Processing in Sensor Networks*, pages 301–316. Springer, 2003.

[22] 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 Trans. on Robotics*, 32(6):1309–1332, 2016.

[23] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2011.

[24] Ricardo Campa and Hussein De La Torre. Pose control of robot manipulators using different orientation representations: A comparative review. In *Proc. American Control Conf.*, pages 2855–2860. IEEE, 2009.

[25] Jason Campbell, Rahul Sukthankar, Illah Nourbakhsh, and Aroon Pahwa. A robust visual odometry and precipice detection system using consumer-grade monocular vision. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3421–3427, 2005.

[26] Patrick JF Carle, Paul T Furgale, and Timothy D Barfoot. Long-range rover localization by matching lidar scans to orbital elevation maps. *J. of Field Robotics*, 27(3):344–370, 2010.

[27] Carsurance. Car Accident Statistics in The U.S. 2020 Update. https://carsurance.net/blog/car-accident-statistics/ 2018.

[28] Andrea Censi. An ICP variant using a point-to-line metric. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 19–25, 2008.

[29] Google Ceres. Google Ceres Solver. http://ceres-solver.org.

[30] Brigitte d'Andréa Novel, Guy Campion, and Georges Bastin. Control of nonholonomic wheeled mobile robots by state feedback linearization. *The Int. J. of Robotics Research*, 14(6):543–559, 1995.

[31] Mohamed A Daoud, Mostafa Osman, Mohamed W Mehrez, and William W Melek. Path-following and adjustable driving behavior of autonomous vehicles using dual-objective nonlinear mpc. In *Proc. IEEE Int. Conf. of Vehicular Electronics and Safety (ICVES)*, pages 1–6, 2019.

[32] Hanieh Deilamsalehy and Timothy C Havens. Sensor fused three-dimensional localization using imu, camera and lidar. In *Proc. IEEE SENSORS*, pages 1–3, 2016.

[33] Alok Desai and Dah-Jye Lee. Visual odometry drift reduction using syba descriptor and feature transformation. *IEEE Trans. on Intelligent Transportation Systems*, 17(7):1839–1851, 2016.

[34] Paulo SR Diniz. *Adaptive filtering.* Springer, 1997.

[35] Nabil M Drawil, Haitham M Amar, and Otman A Basir. Gps localization accuracy classification: A context-based approach. *IEEE Trans. on Intelligent Transportation Systems*, 14(1):262–273, 2012.

[36] P. Duan, G. Tian, and H. Wu. A multi-sensor-based mobile robot localization framework. In *Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO 2014)*, pages 642–647, 2014.

[37] Rodolphe Dubois, Sylvain Bertrand, and Alexandre Eudes. Performance evaluation of a moving horizon estimator for multi-rate sensor fusion with time-delayed measurements. In *Proc. 22nd Int. Conf. on System Theory, Control and Computing (ICSTCC)*, pages 664–669. IEEE, 2018.

[38] Jos Elfring, Rein Appeldoorn, Sjoerd van den Dries, and Maurice Kwakkernaat. Effective world modeling: Multisensor data fusion methodology for automated driving. *Sensors*, 16(10):1668, 2016.

[39] J. Engel, J. Stckler, and D. Cremers. Large-scale direct slam with stereo cameras. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1935–1942, 2015.

[40] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 1449–1456, 2013.

[41] Jorge Esparza-Jiménez, Michel Devy, and José Gordillo. Visual EKF-SLAM from heterogeneous landmarks. *Sensors*, 16(4):489, 2016.

[42] Hanqi Fan and Shuai Zhang. Stereo odometry based on careful frame selection. In *Proc. 10th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 177–180. IEEE, 2017.

[43] Timm Faulwasser, Tobias Weber, Pablo Zometa, and Rolf Findeisen. Implementation of nonlinear model predictive path-following control for an industrial robot. *IEEE Trans. on Control Systems Technology*, 25(4):1505–1511, 2016.

[44] Bo Feng, Mengyin Fu, Hongbin Ma, Yuanqing Xia, and Bo Wang. Kalman filter with recursive covariance estimation - sequentially estimating process noise covariance. *IEEE Trans. on Industrial Electronics*, 61(11):6253–6263, 2014.

[45] Sheng Feng, Shigen Shen, Longjun Huang, Adam C Champion, Shui Yu, Chengdong Wu, and Yunzhou Zhang. Three-dimensional robot localization using cameras in wireless multimedia sensor networks. *J. of Network and Computer Applications*, 146:102425, 2019.

[46] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[47] Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90, 2012.

[48] Yasutaka Furukawa, Carlos Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[49] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[50] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[51] Agneev Guin and Akshay Kumar Burusa. Indoor robot tracking using wi-fi routers by monte carlo localization.

[52] Takoi K Hamrita, EW Tollner, and Robert L Schafer. Toward fulfilling the robotic farming vision: Advances in sensors and controllers for agricultural applications. *IEEE Trans. on Industry Applications*, 36(4):1026–1032, 2000.

[53] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[54] Christoph Helmberg, Franz Rendl, Robert J Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM J. on Optimization*, 6(2):342–361, 1996.

[55] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The Int. J. of Robotics Research*, 31(5):647–663, 2012.

[56] HEXAGON. Real-time Kinematic. https://novatel.com/an-introduction-to-gnss/chapter-5-resolving-errors/real-time-kinematic-rtk.

[57] Wen-Tsai Huang, Chun-Lung Tsai, and Huei-Yung Lin. Mobile robot localization using ceiling landmarks and images captured from an rgb-d camera. In *Proc. IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, pages 855–860, 2012.

[58] Yunliang Jiang, Yunxi Xu, and Yong Liu. Performance evaluation of feature detection and matching in stereo visual odometry. *Neurocomputing*, 120:380–390, 2013.

[59] Simon Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[60] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Proc. Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–194. International Society for Optics and Photonics, 1997.

[61] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. *J. of Basic Engineering*, 83(1):95–108, 1961.

[62] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *J. of basic Engineering*, 82(1):35–45, 1960.

[63] Jonathan Kelly and Gaurav S Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The Int. J. of Robotics Research*, 30(1):56–79, 2011.

[64] Pyojin Kim, Brian Coltin, and H Jin Kim. Low-drift visual odometry in structured environments by decoupling rotational and translational motion. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 7247–7253, 2018.

[65] Pyojin Kim, Brian Coltin, and Hyoun Jin Kim. Visual odometry with drift-free rotation estimation using indoor scene regularities. In *BMVC*, volume 2, page 7, 2017.

[66] Kazuki Kimura, Yutaro Hiromachi, Kenichiro Nonaka, and Kazuma Sekiguchi. Vehicle localization by sensor fusion of lrs measurement and odometry information based on moving horizon estimation. In *Proc. IEEE Conf. on Control Applications (CCA)*, pages 1306–1311, 2014.

[67] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 486–492, 2010.

[68] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154, 2004.

[69] Yasmine Koubaa, Mohamed Boukattaya, and Tarak Damak. Robust control of wheeled mobile robot in presence of disturbances and uncertainties. In *Proc. 14th Int. Conf. on Sciences and Techniques of Automatic Control & Computer Engineering-STA'2013*, pages 274–280. IEEE, 2013.

[70] Tomas Krejci. kitti2bag, [Last Accessed 2020-02-01].

[71] Aravindh Krishnamoorthy and Deepak Menon. Matrix inversion using cholesky decomposition. In *Proc. Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 70–72. IEEE, 2013.

[72] R. Kmmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3607–3613, 2011.

[73] Khaoula Lassoued, Isabelle Fantoni, and Philippe Bonnifait. Mutual localization and positioning of vehicles sharing gnss pseudoranges: Sequential bayesian approach and experiments. In *Proc. IEEE 18th Int. Conf. on Intelligent Transportation Systems*, pages 1896–1901. IEEE, 2015.

[74] CKM Lee. Development of an industrial internet of things (iiot) based smart robotic warehouse management system. In *Proc. Int. Conf. on Information Resources Management (CONF-IRM)*. Association For Information Systems, 2018.

[75] Hyukjung Lee, Joohwan Chun, Kyeongjin Jeon, and Heedeok Lee. Efficient ekf-slam algorithm based on measurement clustering and real data simulations. In *Proc. IEEE 88th Vehicular Technology Conf. (VTC-Fall)*, pages 1–5, 2019.

[76] Kooktae Lee, Woojin Chung, and Kwanghyun Yoo. Kinematic parameter calibration of a car-like mobile robot to improve odometry accuracy. *Mechatronics*, 20(5):582–595, 2010.

[77] Sooyong Lee and Jae-Bok Song. Robust mobile robot localization using optical flow sensors and encoders. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 1039–1044, 2004.

[78] Young-Hee Lee, Chen Zhu, Gabriele Giorgi, and Christoph Günther. Fusion of monocular vision and radio-based ranging for global scale estimation and drift mitigation. *arXiv preprint arXiv:1810.01346*, 2018.

[79] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 7286–7291, 2018.

[80] Wenling Li, Wenhao Zheng, and Yang Liu. Indoor localization for mobile robots using odometry and vision system: A pseudo measurement approach. In *Proc. 5th Int. Conf. on Information, Cybernetics, and Computational Social Systems (ICCSS)*, pages 457–462. IEEE, 2018.

[81] Xianlong Li and Chongyang Zhang. Robust rgb-d visual odometry based on the line intersection structure feature in low-textured scenes. In *Proc. IEEE Int. Conf. on Cloud Computing and Intelligence Systems (CCIS)*, pages 390–394, 2018.

[82] Yi Li, Li He, Xiang Zhang, Lei Zhu, Hong Zhang, and Yisheng Guan. Multi-sensor fusion localization of indoor mobile robot. In *Proc. IEEE Int. Conf. on Real-time Computing and Robotics (RCAR)*, pages 481–486, 2019.

[83] Andong Liu, Wen-An Zhang, Michael ZQ Chen, and Li Yu. Moving horizon estimation for mobile robots with multirate sampling. *IEEE Trans. on Industrial Electronics*, 64(2):1457–1467, 2017.

[84] Wenwen Liu, Yuanchang Liu, and Richard Bucknall. A robust localization method for unmanned surface vehicle (usv) navigation using fuzzy adaptive kalman filtering. *IEEE Access*, 7:46071–46083, 2019.

[85] Yuxiang Liu, Lin Zuo, Changhua Zhang, and Fenglian Liu. Fast and accurate robot localization through multi-layer pose correction. In *Proc. IEEE Int. Conf. on Mechatronics and Automation (ICMA)*, pages 2487–2492, 2019.

[86] David G Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Computer Vision*, 60(2):91–110, 2004.

[87] Wenhuan Lu, Ju Zhang, Xinli Zhao, Jianrong Wang, and Jianwu Dang. Multimodal sensory fusion for soccer robot self-localization based on long short-term memory recurrent neural network. *J. of Ambient Intelligence and Humanized Computing*, 8(6):885–893, Nov 2017.

[88] R. C. Luo and C. C. Chang. Multisensor fusion and integration: A review on approaches and its applications in mechatronics. *IEEE Trans. on Industrial Informatics*, 8(1):49–60, 2012.

[89] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, New York, NY, USA, 1st edition, 2017.

[90] Carlos Eduardo Setenareski Magrin and Eduardo Todt. Hierarchical sensor fusion method based on fingerprint knn and fuzzy features weighting for indoor localization of a mobile robot platform. In *Proc. XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, pages 305–310. IEEE, 2016.

[91] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. National Institute of Science of India, 1936.

[92] Raman Mehra. On the identification of variances and adaptive kalman filtering. *IEEE Trans. on Automatic Control*, 15(2):175–184, 1970.

[93] Raman Mehra. Approaches to adaptive filtering. *IEEE Trans. on Automatic Control*, 17(5):693–698, 1972.

[94] Mohamed W Mehrez, George KI Mann, and Raymond G Gosine. Nonlinear moving horizon state estimation for multi-robot relative localization. In *Proc. IEEE 27th Canadian Conf. on Electrical and Computer Engineering (CCECE)*, pages 1–5, 2014.

[95] Mohamed W. Mehrez, Karl Worthmann, Joseph P.V. Cenerini, Mostafa Osman, William W. Melek, and Soo Jeon. Model predictive control without terminal constraints or costs for holonomic mobile robots. *Robotics and Autonomous Systems*, 127:103468, 2020.

[96] Vicente Milanés, Steven E Shladover, John Spring, Christopher Nowakowski, Hiroshi Kawazoe, and Masahide Nakamura. Cooperative adaptive cruise control in real traffic situations. *IEEE Trans. on Intelligent Transportation Systems*, 15(1):296–305, 2013.

[97] S Mishra, PS Londhe, S Mohan, SK Vishvakarma, and BM Patre. Robust task-space motion control of a mobile manipulator using a nonlinear control with an uncertainty estimator. *Computers & Electrical Engineering*, 67:729–740, 2018.

[98] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proc. 13th Int. Conf. on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[99] Wei Mou, Han Wang, and Gerald Seet. Efficient visual odometry estimation using stereo camera. In *Proc. 11th IEEE Int. Conf. on Control & Automation (ICCA)*, pages 1399–1403, 2014.

[100] David M Mount, Nathan S Netanyahu, and Jacqueline Le Moigne. Efficient algorithms for robust feature matching. *Pattern Recognition*, 32(1):17–38, 1999.

[101] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Trans. on Robotics*, 31(5):1147–1163, 2015.

[102] Derradji Nada, Mounir Bousbia-Salah, and Maamar Bettayeb. Multi-sensor data fusion for wheelchair position estimation with unscented kalman filter. *Int. J. of Automation and Computing*, 15(2):207–217, 2018.

[103] Tayyab Naseer, Benjamin Suger, Michael Ruhnke, and Wolfram Burgard. Vision-based markov localization for long-term autonomy. *Robotics and Autonomous Systems*, 89:147–157, 2017.

[104] David Nister. An efficient solution to the five-point relative pose problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[105] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–I, 2004.

[106] Mostafa Osman. Moving Horizon Estimation Localization. https://github.com/MostafaOsman144/moving_horizon_estimation_localization, June 2012.

[107] Mostafa Osman. DCE: Online Drift Covariance Estimation. https://github.com/lsi-uc3m/covariance_estimation, March 2018.

[108] Mostafa Osman, Ricardo Alonso, Ahmed Hammam, Francisco Miguel Moreno, Abdulla Al-Kaff, and Ahmed Hussein. Multisensor fusion localization using extended h-infinity filter using pre-filtered sensors measurements. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 1139–1144. IEEE, 2019.

[109] Mostafa Osman, Ahmed Hussein, and Abdulla Al-Kaff. Intelligent vehicles localization approaches between estimation and information: A review. In *Proc. IEEE Int. Conf. of Vehicular Electronics and Safety (ICVES)*, pages 1–8, 2019.

[110] Mostafa Osman, Ahmed Hussein, Abdulla Al-Kaff, Fernando García, and José María Armingol. Online adaptive covariance estimation approach for multiple odometry sensors fusion. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 355–360, 2018.

[111] Mostafa Osman, Ahmed Hussein, Abdulla Al-Kaff, Fernando García, and Dongpu Cao. A novel online approach for drift covariance estimation of odometries used in intelligent vehicle localization. *Sensors*, 19(23):5178, 2019.

[112] Mostafa Osman, Mohamed W. Mehrez, Shiyi Yang, Soo Jeon, and William Melek. End-effector stabilization of a 10-dof mobile manipulator using nonlinear model predictive control. In *Proc. IFAC World Congress*, pages 1–6, 2020.

[113] Bhavik Patel, Ya-Jun Pan, and Usman Ahmad. Adaptive backstepping control approach for the trajectory tracking of mobile manipulators. In *Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, pages 1769–1774, 2017.

[114] Valentin Peretroukhin, Lee Clement, and Jonathan Kelly. Reducing drift in visual odometry by inferring sun direction using a bayesian convolutional neural network. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2035–2042, 2017.

[115] Farzin Piltan, MohammadHossain Yarmahmoudi, Mina Mirzaie, Sara Emamzadeh, and Zahra Hivand. Design novel fuzzy robust feedback linearization control with application to robot manipulator. *Int. J. of Intelligent Systems and Applications*, 5(5):1, 2013.

[116] Pixhawk. Pixhawk Auto-pilot. https://pixhawk.org/.

[117] Farzad Pourboghrat and Mattias P Karlsson. Adaptive control of dynamic mobile robots with nonholonomic constraints. *Computers & Electrical Engineering*, 28(4):241–253, 2002.

[118] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[119] Eric B Quist, Peter C Niedfeldt, and Randal W Beard. Radar odometry with recursive-ransac. *IEEE Trans. on Aerospace and Electronic Systems*, 52(4):1618–1630, 2016.

[120] Rajeev Ramanath, Wesley E Snyder, Youngjun Yoo, and Mark S Drew. Color image processing pipeline. *IEEE Signal Processing Magazine*, 22(1):34–43, 2005.

[121] Christopher V Rao, James B Rawlings, and David Q Mayne. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *IEEE Trans. on Automatic Control*, 48(2):246–258, 2003.

[122] Punit Rathore, Dheeraj Kumar, Sutharshan Rajasegarar, and Marimuthu Palaniswami. Maximum entropy-based auto drift correction using high-and low-precision sensors. *ACM Trans. on Sensor Networks (TOSN)*, 13(3):24, 2017.

[123] Punit Rathore, Dheeraj Kumar, Sutharshan Rajasegarar, and Marimuthu Palaniswami. Bayesian maximum entropy and interacting multiple model based automatic sensor drift detection and correction in an iot environment. In *Proc. IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 598–603, 2018.

[124] Ali M Reza. Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement. *J. of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 38(1):35–44, 2004.

[125] Roboteq. Line Following Sensors. https://www.roboteq.com/all-products/magnetic-guide-sensors, 2018.

[126] Robotnik. SUMMIT- XL STEEL Datasheet. www.robotnik.eu, 2018.

[127] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proc. European Conf. on Computer Vision*, pages 430–443. Springer, 2006.

[128] Peter J Rousseeuw. Least median of squares regression. *J. of the American Statistical Association*, 79(388):871–880, 1984.

[129] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proc. Int. Conference on Computer Vision*, pages 2564–2571. IEEE, 2011.

[130] Alexander Rudolph. Quantification and estimation of differential odometry errors in mobile robotics with redundant sensor information. *The Int. J. of Robotics Research*, 22(2):117–128, 2003.

[131] Mohamed Sabry, Abdulla Al-Kaff, Ahmed Hussein, and Slim Abdennadher. Ground vehicle monocular visual odometry. In *Proc. IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3587–3592, 2019.

[132] Mohamed Walid Mehrez Said, George KI Mann, Raymond G Gosine, and Tariq Iqbal. A receding horizon approach for control and state estimation in nonholonomic mobile robots: Stability, and relative localization. 2015.

[133] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics & Automation Magazine*, 18(4):80–92, 2011.

[134] Davide Scaramuzza, Friedrich Fraundorfer, and Roland Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 4293–4299, 2009.

[135] David CH Shaw and Judy ZZ Shaw. Vehicle collision avoidance system, June 25 1996. US Patent 5,529,138.

[136] Jae Hong Shim and Young Im Cho. A mobile robot localization via indoor fixed remote surveillance cameras. *Sensors*, 16(2):195, 2016.

[137] Jia-Ming Shyu and Ching-Wang Chuang. Automatic parking device for automobile, June 5 1990. US Patent 4,931,930.

[138] Roland Siegwart, Illah Reza Nourbakhsh, Davide Scaramuzza, and Ronald C Arkin. *Introduction to autonomous mobile robots*. MIT press, 2011.

[139] Frederico Fernandes Afonso Silva and Bruno Vilhena Adorno. Whole-body control of a mobile manipulator using feedback linearization based on dual quaternions. In *Proc. XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, pages 293–298. IEEE, 2016.

[140] D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006.

[141] Andrea Simonetto, Daniele Balzaretti, and Tamás Keviczky. A distributed moving horizon estimator for mobile robot localization problems. *IFAC Proceedings Volumes*, 44(1):8902–8907, 2011.

[142] J-JE Slotine and Li Weiping. Adaptive manipulator control: A case study. *IEEE Trans. on Automatic Control*, 33(11):995–1003, 1988.

[143] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. Real-time visual odometry from dense rgb-d images. In *Proc. IEEE Int. Conf. on Computer Vision Workshops (ICCV Workshops)*, pages 719–722, 2011.

[144] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 573–580, 2012.

[145] Jin Sun, Xiaosu Xu, Yiting Liu, Tao Zhang, and Yao Li. Fog random drift signal denoising based on the improved ar model and modified sage-husa adaptive kalman filter. *Sensors*, 16(7):1073, 2016.

[146] Tim Tang, David Yoon, François Pomerleau, and Timothy D Barfoot. Learning a bias correction for lidar-only motion estimation. In *Proc. 15th Conf. on Computer and Robot Vision (CRV)*, pages 166–173, 2018.

[147] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[148] Yun-Peng Tian, Xiao-Jun Yang, Yun-Zeng Guo, and Feng Liu. Filtering and analysis on the random drift of fog. In *Proc. Optical Fiber Sensors and Applications (AOPC)*, volume 9679, page 96790J. International Society for Optics and Photonics, 2015.

[149] Masahiro Tomono. 3-D localization and mapping using a single camera based on structure-from-motion with automatic baseline selection. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3342–3347, 2005.

[150] Ching-Chih Tsai. A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements. *IEEE Trans. on Instrumentation and Measurement*, 47(5):1399–1404, 1998.

[151] TUM. Submission form for automatic evaluation of rgb-d slam results. https://vision.in.tum.de/data/datasets/rgbd-dataset/online_evaluation.

[152] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *J. of Field Robotics*, 25(8):425–466, 2008.

[153] Bart Van Arem, Cornelie JG Van Driel, and Ruben Visser. The impact of cooperative adaptive cruise control on traffic-flow characteristics. *IEEE Trans. on Intelligent Transportation Systems*, 7(4):429–436, 2006.

[154] Velodyne. Velodyne LiDAR Sensor. https://velodynelidar.com/.

[155] VICON. VICON motion capture system.

[156] H-J Von Der Hardt, Didier Wolf, and René Husson. The dead reckoning localization system of the wheeled mobile robot romane. In *Proc. IEEE/SICE/RSJ Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pages 603–610, 1996.

[157] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006.

[158] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proc. IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, 2000.

[159] Hairong Wang, Zhihong Deng, Bo Feng, Hongbin Ma, and Yuanqing Xia. An adaptive kalman filter estimating process noise covariance. *Neurocomputing*, 223:12–17, 2017.

[160] Liang Wang, Yihuan Zhang, and Jun Wang. Map-based localization method for autonomous vehicles using 3d-lidar. *IFAC-PapersOnLine*, 50(1):276–281, 2017.

[161] Yuzhi Wang, Anqi Yang, Xiaoming Chen, Pengjun Wang, Yu Wang, and Huazhong Yang. A deep learning approach for blind drift calibration of sensor networks. *IEEE Sensors J.*, 17(13):4158–4171, 2017.

[162] Yuzhi Wang, Anqi Yang, Zhan Li, Pengjun Wang, and Huazhong Yang. Blind drift calibration of sensor networks using signal space projection and kalman filter. In *Proc. IEEE 10th Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, 2015.

[163] Eric W. Weisstein. Angular distance" from mathworld – a wolfram web resource. https://mathworld.wolfram.com/AngularDistance.html.

[164] Eric W. Weisstein. Law of cosines" from mathworld – a wolfram web resource. https://mathworld.wolfram.com/LawofCosines.html.

[165] Shuhuan Wen, Xiao Chen, Chunli Ma, Hak-Keung Lam, and Shaoyang Hua. The q-learning obstacle avoidance algorithm based on ekf-slam for nao autonomous walking under unknown environments. *Robotics and Autonomous Systems*, 72:29–36, 2015.

[166] Louis Whitcomb, Dana R Yoerger, Hanumant Singh, and Jonathan Howland. Advances in underwater robot vehicles for deep ocean exploration: Navigation, control, and survey operations. In *Robotics Research*, pages 439–448. Springer, 2000.

[167] Karl Worthmann, Mohamed W Mehrez, Mario Zanon, George KI Mann, Raymond G Gosine, and Moritz Diehl. Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs. *IEEE Transactions on Control Systems Technology*, 24(4):1394–1406, 2015.

[168] Meiqing Wu, Siew-Kei Lam, and Thambipillai Srikanthan. A framework for fast and robust visual odometry. *IEEE Trans. on Intelligent Transportation Systems*, 18(12):3433–3448, 2017.

[169] Zhi Yan, Li Sun, Tomas Krajnik, and Yassine Ruichek. EU long-term dataset with multiple sensors for autonomous driving. *CoRR*, abs/1909.03330, 2019.

[170] Nan Yang, Rui Wang, Xiang Gao, and Daniel Cremers. Challenges in monocular visual odometry photometric calibration, motion bias, and rolling shutter effect. *IEEE Robotics and Automation Letters*, 3(4):2878–2885, 2018.

[171] Wenkao Yang and Xiangwei Zhai. Contrast limited adaptive histogram equalization for an advanced stereo visual slam system. In *Proc. IEEE Int. Conf on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 131–134, 2019.

[172] Setareh Yazdkhasti and Jurek Z Sasiadek. Multi sensor fusion based on adaptive kalman filtering. In *Advances in Aerospace Guidance, Navigation and Control*, pages 317–333. Springer, 2018.

[173] Jingang Yi, Junjie Zhang, Dezhen Song, and Suhada Jayasuriya. Imu-based localization and slip estimation for skid-steered mobile robots. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2845–2850, 2007.

[174] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual

odometry with deep feature reconstruction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 340–349, 2018.

[175] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Proc. Robotics: Science and Systems*, volume 2, 2014.

[176] Ji Zhang and Sanjiv Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416, 2017.

[177] Ji Zhang and Sanjiv Singh. Laservisualinertial odometry and mapping with high robustness and low drift. *J. of Field Robotics*, 35(8):1242–1264, 2018.

[178] Jun Zhang, Viorela Ila, and Laurent Kneip. Robust visual odometry in underwater environment. In *Proc. OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, pages 1–9, 2018.

[179] Yu-feng Zhang, Qi-xun Zhou, Ju-zhong Zhang, Yi Jiang, and Kai Wang. A fast-slam algorithm based on nonlinear adaptive square root unscented kalman filter. *Mathematical Problems in Engineering*, 2017, 2017.

[180] Xiangmo Zhao, Haigen Min, Zhigang Xu, Xia Wu, Xiaochi Li, and Pengpeng Sun. Image antiblurring and statistic filter of feature space displacement: application to visual odometry for outdoor ground vehicle. *J. of Sensors*, 2018, 2018.

[181] Bo Zhou, Zhongqiang Tang, Kun Qian, Fang Fang, and Xudong Ma. A lidar odometry for outdoor mobile robots using ndt based scan matching in gps-denied environments. In *Proc. IEEE 7th Annual Int. Conf. on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 1230–1235, 2017.

[182] Dingfu Zhou, Yuchao Dai, and Hongdong Li. Reliable scale estimation and correction for monocular visual odometry. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 490–495, 2016.

[183] Dingfu Zhou, Yuchao Dai, and Hongdong Li. Ground-plane-based absolute scale estimation for monocular visual odometry. *IEEE Trans. on Intelligent Transportation Systems*, 2019.

[184] Karel Zuiderveld. Contrast limited adaptive histogram equalization. In *Graphics gems IV*, pages 474–485. Academic Press Professional, Inc., 1994.