A Novel Filtering Approach in Visual Odometry for Autonomous Ground Vehicles Application

A NOVEL FILTERING APPROACH IN VISUAL ODOMETRY FOR AUTONOMOUS GROUND VEHICLES APPLICATION

BY

SARA ALIZADEH, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2017) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE: A Novel Filtering Approach in Visual Odometry for Autonomous Ground Vehicles Application

AUTHOR: Sara Alizadeh

B.Sc., (Electrical Engineering)

Amirkabir University of Technology, Tehran, Iran

SUPERVISOR: Dr. T. Kirubarajan

NUMBER OF PAGES: xi, 96

To my mother and my father.

Abstract

A monocular Visual Odometry system has been developed and tested on different datasets and the outputs have been compared with the available ground truth information to analyze the precision of the system. This system is capable of estimating the 3D position of a ground vehicle robustly and in real time.

One of the main challenges of monocular VO is the ambiguity of the scale estimation which is addressed by assuming that the ground is locally planar and the height of the mounted camera from the ground is fixed and known.

In order to improve the VO estimation and to help other stages of VO process an effective filtering approach is utilized. It is shown that an IMM filtering can address the needs of this specific application, as the movement of a ground vehicle is different depending on different scenarios.

The results of simulation on the well-known KITTI dataset demonstrates that our system's accuracy improved compared to what is considered to be one of the best state-of-the-art monocular Visual Odometry system.

Acknowledgements

I would like to take the opportunity to thank some very special people; without their help I could not do any of this.

First and foremost, I would like to express my deepest gratitude to my academic supervisor, Dr. T. Kirubarajan whose expert advice and guidance, motivated me throughout my research. I would like to thank him for all the supports he gave me for the past couple of years.

I am most grateful to Dr. R. Tharmarasa, for his much needed guidance and inputs which greatly improved my research. I would also like to thank the faculty of ECE department especially Dr. J. Reilly and Dr. T. Davidson. I learnt a lot from these great people. Thanks to Cheryl Gies the graduate administrative assistant of ECE department.

Last but by no means least, I would like to express the profound gratitude from my deep heart to my father and my mother again, for which my mere expression of thanks likewise does not suffice. Without their unconditional love and consistent support through all the years, all my achievements have simply been impossible. And here goes my special thanks to my sister, Maryam for being always not only a sister but being a true friend.

Notation and abbreviations

ADAS: Advanced Driver Assistant Systems

VO: Visual Odometry

SFM: Structure from Motion

RANSAC: RANdom SAmple Consensus

DoF: Degree of Freedom

SLAM: Simultaneous Localization and Mapping

V-SLAM: Visual Simultaneous Localization and Mapping

PnP: Perspective-n-point-Problem

KF: Kalman Filter

EKF: Extended Kalman Filter

IMM: Interacting Multiple Model

SPD: Symmetric Positive Definite

DLT: Direct Linear Transform

SVD: Singular Value Decomposition

SAD: Sum of Absolute Differences

UAV: Unmanned Aerial Vehicle

Contents

List of Figures

Chapter 1

Introduction and Motivation

Improving autonomous systems that can facilitate everyday tasks is a major challenge in modern technological era. As such autonomous vehicles are of intense focus in automotive industry. Autonomous driving and Advanced Driver Assistant systems (ADAS) can help reduce human errors and the resulting traffic fatality.

To move the vehicle autonomously and securely the vehicle needs to be aware of the ego motion, its 3D position changing in a static environment, in addition to prospective obstacles in the surrounding. Odometry is the process of estimating the position and orientation of an agent (e.g., vehicle, human, and robot) relative to its initial location [35]. For autonomous navigation, obstacle detection and avoidance, motion tracking and many other application, an agent must know its position over time. Vision based odometry or Visual Odometry (VO) is a technique used to estimate the ego motion of an agent using only the input of a single or multiple cameras attached to it [35].

Compared to other sensor-based localization systems, cameras are more reliable and robust [33]. Detection of road edges, lanes, pedestrians, other vehicles and their

movement are crucial for autonomous vehicles and Advanced Driver Assistant systems. The images captured by a camera can provide a large amount of meaningful information to be used for such detections and many other purposes which makes the utilization of a camera more beneficial than other sensors. Furthermore, compared with other localization sensors, optical cameras are cost efficient. Optical cameras are also, passive, so they do not have the problem of interference with other active sensors like LIDAR or RADAR. Cameras are small, lightweight, low powered and flexible (they can be employed in any vehicle e.g. air, land, underwater) [1].

1.1 Literature Review

The problem of retrieving camera poses and 3D structure from a set of images is called structure from motion (SFM) in the computer vision science. For further study of the earliest work in this area, readers can refer to [13] and [23]. Structure from motion is more general which Visual Odometry is a specific case of it. SFM focuses on the 3D reconstruction of the structure and camera poses simultaneously from a set of ordered or unordered images and utilizes an off-line optimization to refine the final structure and camera poses. On the other hand, VO concentrates on recovering the 3D motion of the camera sequentially and in real time [35].

The problem of vehicle's ego motion estimation from only the visual inputs is introduced in [28]. The earliest research in VO [25], [24] and [21] was done for the NASA Mars exploration program in order to produce rovers with the ability to determine their 6 degree of freedom (DoF) motion in bumpy and rough terrains.

Stereo VO

The research that has been done in VO thus for mostly utilizes stereo cameras. In addition to the work of Moravec [28], Matthies and Shafer [25], [24] used a stereo system. Compared to the work in [28], which uses a scalar representation of the uncertainty, they utilize an error covariance matrix of the triangulated features, which outperforms Moravec's proposed algorithm [35].

The methods so far triangulate the 3D points for every stereo pair, and the relative motion is calculated as a 3D-to-3D point registration (alignment) problem [35]. A different method by Nister et al. [32] is not only the first real-time long-run robust implementation but also the paper that proposed the term VO for the first time. In this paper, instead of calculating the relative motion as a 3D-to-3D point registration problem they utilized 3D-to-2D pose estimation method [35].

Monocular VO

The works in Visual Odometry using one camera can be categorized in feature based approaches [32], [43], [37] and appearance based methods [11], [38], and [27]. The first real time monocular VO was proposed by Nister et al. [32]. It utilized RANSAC for outlier removal and 3D-to-2D camera pose estimation to recover camera pose using a five point minimal solver [31] to compute the hypotheses in RANSAC.

In particular, some works in VO for vehicles assume motion constraints, which lower the computational complexity and increase motion accuracy. For example, the works in [22], [15], [44] and [12] assume homography constraint for the ego motion estimation. Scaramuzza et al. [37], [34] proposed a one point RANSAC outliers removal based on the vehicle nonholonomic constraints to speed up ego motion estimation.

Their future work indicates that nonholonomic constraints allow the absolute scale to be retrieved from a single camera when the vehicle turns [36].

Among a few monocular VO systems that have obtained good accuracy in localization mentionable works include [6], [17], [18], but they work for small indoor applications. The challenge for large-scale applications is the problem of scale drift [45]. A well-known technique to tackle this problem of scale drift in SLAM is loop closure [41]. However, loop closer is an expensive approach as it maintains the global information of trajectory. Furthermore, for autonomous vehicle application there is no guarantee that the vehicle encounters loops.

One of the best state-of-the-art on-line monocular VO is the work in [10]. This system computes relative pose between every neighboring frame by extracting 2D-to-2D matching through a fundamental matrix estimation. It further calculates the relative scale assuming a locally planar ground. This system is the counterpoint system throughout this research.

V-SLAM (Visual Simultaneous Localization and Mapping)

The aim of SLAM (Simultaneous Localization and Mapping) is to estimate the robot path consistently and keep track of a map of the surrounding to notice when the robot gets back to a visited area [35]. VO only deals with local consistency of the trajectory and the local map is used to get a more accurate local trajectory, whereas, SLAM deals with the global map consistency. In other words, VO can be utilized as a building block for a complete SLAM block diagram. Even though the global consistency of the camera path is sometimes advisable, VO trades off consistency for real time performance [35].

Among the early works in real time V-SLAM, work of Civera et al. [4] proposed a one point RANSAC with combination of the Kalman filter that took advantage of the available prior information from the filter in the RANSAC model hypothesis step. Strasdat et al. [41] prosed an algorithm for large scale V-SLAM that utilizes the key frame optimization [17]. Most of these works have been restricted to small, indoor areas but a few recent works [5], [4] and [26] are used for large scale workspaces.

1.2 Contribution

Most of the works that has been done in Visual Odometry mainly focused on the computer vision aspects of VO. Some research of feature extraction are [32], [43], [37], [11], [38], and [27]. Outstanding works in scale estimation for Monocular VO can be found in [41], [45], [16] and [39]. Utilizing extensive validation and refinement mechanisms through bundle adjustments and multi-threaded design can be found in [40].

In this research after extracting and matching features, the relative pose of consecutive frames is estimated through a fundamental matrix estimation and scale calculation assuming a locally planar ground. The main focus is to find a Kalman filter approach to improve the VO estimation and to help other stages of VO process. By applying an efficient Kalman filter, better results can be obtained with the same amount of computational complexity. This filtering also can be applied in any other aspects when we are dealing with a moving camera and we are interested in the ego motion and the own displacement of the camera.

On top of the estimation procedure we first place the simplest case, the standard Kalman filter, assuming constant acceleration. By analyzing the measurement and

simulated results, it has been seen that a simple Kalman filter can not meet our needs. Further, to get better estimation of VO in terms of accuracy, a nonlinear transfer function has been utilized. Finally, an IMM approach is applied with different levels of process noise. A varieties of scenarios have been studied and analyzed on different sequences of the well-known KITTI dataset.

Chapter 2

Background

Computer vision is the science that aims to understand the 3D structure of the surrounding and to give the capability to a machine or computer from digital images or videos.

Regardless of all of the advances in computer vision, the dream of having a computer understand and interpret an image at the same level as a human is ongoing. The vision is an inverse problem; we have inadequate information from which we want to extract some unknowns. Hence, we must use physics-based and probabilistic models to disambiguate between different solutions [42], [14].

In order to analyze and study images we need a vocabulary for describing the geometry of a scene.

2.1 Homogeneous Notation

2D Lines

Let $ax + by + c = 0$ be a line in the plane or equivalently it can be represented by the vector $(a, b, c)^T$. As the lines $ax + by + c = 0$ and $(ka)x + (kb)y + (kc) = 0$, $k \neq 0$, are the same, the correspondence between lines and vectors $(a, b, c)^T$ is not one-to-one. Therefore, the vectors $(a, b, c)^T$ and $k(a, b, c)^T$ describe the same line, for any non-zero k. An equivalence class of vectors under this equivalence relationship is called as a **homogeneous** vector. Any specific vector $(a, b, c)^T$ is a representative of an equivalence class [14].

2D Points

The point $\mathbf{x} = (x, y)^T$ in $\mathbb{R}^2$ can be written as a 3-vector by adding a final element of 1, $\bar{\mathbf{x}} = (x, y, 1)^T$ (augmented vector). Hence, the point $(x, y)^T$ in $\mathbb{R}^2$ is the representation of the set of vectors $(kx, ky, k)^T$ for varying values of k. In other words, a homogeneous vector representative of a point is of the form $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w})^T$, representing the point $(\frac{\tilde{x}}{\tilde{w}}, \frac{\tilde{y}}{\tilde{w}})^T$ in $\mathbb{R}^2$.

$$\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w})^T = \tilde{w}(x, y, 1)^T = \tilde{w}\bar{\mathbf{x}}$$

Points and Lines in Infinity

Ideal points (points at infinity) are the points with last coordinate $\tilde{w} = 0$. The set of all ideal points can be described as $(\tilde{x}, \tilde{y}, 0)^T$. This set lies on the line at infinity, $l_\infty = (0, 0, 1)^T$. These kind of homogeneous vectors correspond to finite points in $\mathbb{R}^2$, which agrees with the idea that parallel lines meet at infinity [14].

Projective Space /Projective Plane

The set of equivalence classes of vectors in $\mathbb{R}^3 - (0,0,0)^T$ outlines the **projective space**, $\mathbb{P}^2$. **Projective geometry** is the study of the geometry of projective space. In other words, $\mathbb{R}^2$ can be expanded by adding points with last element $\tilde{w} = 0$. The resulting space is known as the projective space, $\mathbb{P}^2$, which consists of all homogeneous 3-vectors.

By way of explanation, $\mathbb{P}^2$ is a set of rays in $\mathbb{R}^3$. A ray through the origin is formed by the set of all vectors $k(\tilde{x}, \tilde{y}, \tilde{w})^T$ as k changes. Such a ray shows a single point in $\mathbb{P}^2$. In this model, planes passing through the origin are the lines in $\mathbb{P}^2$.

As an example, see Figure 2.1. Points and lines are acquired by intersecting this set of rays and planes by the plane $x_3 = 1$. Lines in the x_1x_2-plane and the x_1x_2-plane are representing ideal points and l_∞ respectively [14].

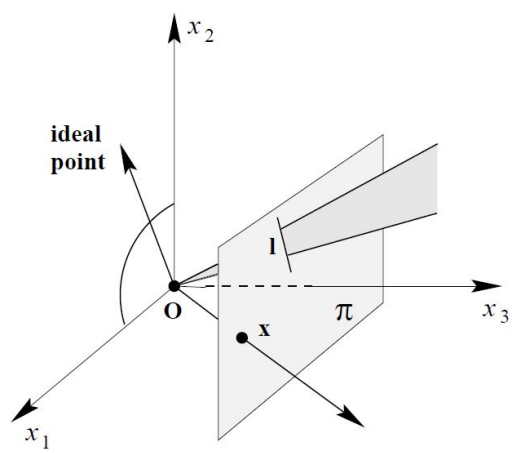

Figure 2.1: A model of the projective plane [14].

2.2 Transformations

Some of the simplest 2D planar transformations are briefly explained in the following sections.

Euclidean/ Rotation+Translation/ Rigid Body Motion

It can be expressed as $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$ or

$$\mathbf{x}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}}$$

where

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

is an orthonormal rotation matrix with $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $|\mathbf{R}| = 1$

Affine

$\mathbf{x}' = \mathbf{A}\bar{\mathbf{x}}$, is described the affine transformation where $\mathbf{A}$ is a 2×3 matrix, i.e.,

$$\mathbf{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \bar{\mathbf{x}}$$

Projective/ Perspective Transform /Homography

This transformation functions on homogeneous coordinates,

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{H}}\tilde{\mathbf{x}}$$

where $\tilde{\mathbf{H}}$ is a 3×3 homogeneous matrix, i.e., two $\tilde{\mathbf{H}}$ matrices that is different only by scale are equal. The resulting homogeneous coordinate $\tilde{\mathbf{x}}'$ must be normalized in order to acquire an inhomogeneous result $\mathbf{x}$

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}, y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

All the above transformation can be extended to 3D transformations. The main difference between 2D and 3D transformations is the 3D rotation matrix R, which can be parameterized in different ways [42].

2.2.1 3D Rotation

Euler

Let $\mathbf{x} = (1, 0, 0)^T$, $\mathbf{y} = (0, 1, 0)^T$ and $\mathbf{z} = (0, 0, 1)^T$ denote an orthonormal basis, which can be called principle axis as an axis of rotation. We can show three principle rotations through the respective angles α, β and γ by $\mathbf{R}_x(\alpha)$, $\mathbf{R}_y(\beta)$ and $\mathbf{R}_z(\gamma)$. The combined rotation can be written as

$$\mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma) =$$
$$\begin{bmatrix} \cos\alpha\cos\gamma - \cos\beta\sin\alpha\sin\gamma & -\sin\alpha\cos\gamma - \cos\beta\cos\alpha\sin\gamma & -\sin\beta\sin\gamma \\ \cos\beta\sin\alpha\cos\gamma + \cos\alpha\sin\gamma & \cos\beta\cos\alpha\cos\gamma - \sin\alpha\sin\gamma & -\sin\beta\cos\gamma \\ \sin\beta\sin\alpha & \sin\beta\cos\alpha & \cos\beta \end{bmatrix}$$

Axis Angle

Another way to represent the vector rotation is called axis angle rotation. It is denoted by a unit vector $\mathbf{n}$ and a rotation angle θ. This shows the rotation $\mathbf{R}_n(\theta)$ through angle θ about the axis $\mathbf{n}$; ($\mathbf{n}$, θ).

Quaternion

The quaternions are a 4D vector space, which can be denoted as

$$\mathbf{q} = e_0 + e_1\mathbf{i} + e_2\mathbf{j} + e_3\mathbf{k}$$

The first term e_0 is known as the real part of $\mathbf{q}$ and $e_1\mathbf{i}+e_2\mathbf{j}+e_3\mathbf{k}$ is the imaginary part. An alternative way to show $\mathbf{q}$ is

$$\mathbf{q} = (e_0, \mathbf{e})$$

where $\mathbf{e} = (e_1, e_2, e_3)^T$ shows the imagery part.

The corresponding rotation matrix can be written as:

$$\mathbf{R} = \begin{bmatrix} 1-2(e_2^2+e_3^2) & 2(e_1e_2-e_0e_3) & 2(e_1e_3+e_0e_2) \\ 2(e_1e_2+e_0e_3) & 1-2(e_1^2+e_3^2) & 2(e_2e_3-e_0e_1) \\ 2(e_1e_3-e_0e_2) & 2(e_2e_3+e_0e_1) & 1-2(e_1^2+e_2^2) \end{bmatrix}$$

Here, we shall show the relationship between these three forms of rotation representation. Let's $\mathbf{X} = (x, y, z)^T$ represents the Euler representation in 3D, the equivalent angle-axis representation can be denotes as

$$a =< \mathbf{a}, \alpha >=< \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \alpha >=< \begin{bmatrix} \frac{x}{||\mathbf{X}||} \\ \frac{y}{||\mathbf{X}||} \\ \frac{z}{||\mathbf{X}||} \end{bmatrix}, ||\mathbf{X}|| >$$

Therefore, the quaternion q showing the same rotation can be written as

$$q = (\cos\frac{\alpha}{2}, \frac{a_x}{||\mathbf{a}||}\sin\frac{\alpha}{2}, \frac{a_y}{||\mathbf{a}||}\sin\frac{\alpha}{2}, \frac{a_z}{||\mathbf{a}||}\sin\frac{\alpha}{2})^T$$

2.3 3D to 2D Projections

3D to 2D Projections can be done using a linear 3D to 2D projection matrix. The perspective model is used more commonly, as it models the behavior of cameras more precisely [42].

To get 2D points, the 3D points are divided by their z elements. It can be represented as

$$\bar{\mathbf{x}} = \mathcal{P}_z(\mathbf{p}) = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

Using homogeneous coordinates, it is written as

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{p}}$$

In this projection we discard the w component of $\mathbf{p}$. Therefore, it is impossible to

get the distance of the 3D point from the image after projection.

2.4 Calibration

After a 3D point has been projected to 2D image plane, we still need to transform the 2D point based on the position of sensor plane. In figure 2.2 a 3D point $\mathbf{p}_c$ has been projected onto the sensor planes; 2D point $\mathbf{p}$, where s_x and s_y are the pixel spacings, $\mathbf{c}_s$ is the 3D origin of the sensor plane coordinate system and $\mathbf{O}_c$ is the camera center.

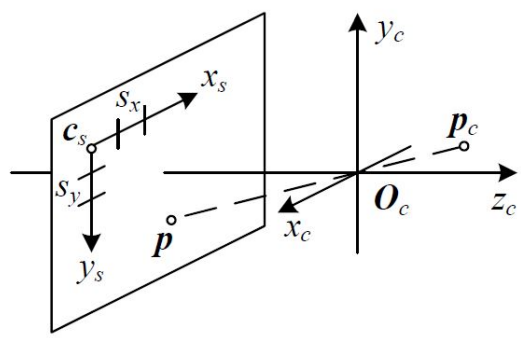

Figure 2.2: Projection of a 3D point onto the sensor plane [42].

In order to map 3D camera-centered points $\mathbf{p}_c$ to 2D sensor plane $\tilde{\mathbf{x}}_s$ the **Calibration matrix**, $\mathbf{K}$, is used.

$$\tilde{\mathbf{x}}_s = \mathbf{K}\mathbf{p}_c$$

When calibrating a camera using a series of measurements, we estimate the intrinsic ($\mathbf{K}$) and extrinsic ($\mathbf{R}$, $\mathbf{t}$) camera parameters at the same time,

$$\tilde{\mathbf{x}}_s = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{p}_w = \mathbf{P}\mathbf{p}_w$$

where $\mathbf{p}_w$ is a 3D world coordinate and

$$\mathbf{P} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$$

or

$$\tilde{\mathbf{P}} = \begin{bmatrix}\mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1\end{bmatrix}\begin{bmatrix}\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1\end{bmatrix} = \tilde{\mathbf{K}}\mathbf{E}$$

is called the **camera matrix**. In this equation $\mathbf{E}$ is a 3D rigid-body (Euclidean) transformation and $\tilde{\mathbf{K}}$ is the calibration matrix.

By having a full 3×4 camera matrix $\mathbf{P} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$, an upper-triangular $\mathbf{K}$ matrix can be calculated by QR factorization.

One possibility to write the matrix $\mathbf{K}$ is

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where (c_x, c_y) indicates the optical center in pixel coordinates , f_x and f_y are the independent focal lengths for the sensor x and y dimensions and s shows skew between the sensor axes when the sensor is not being mounted perpendicular to the optical axis. It is more appropriate to demonstrate the focal length f in pixel, therefore, it can be used easily in the calibration matrix $\mathbf{K}$.

2.5 Mapping from one camera to another

Let $\mathbf{p}$ be a 3D point, which is projected to image plane at the location $\tilde{\mathbf{x}}_\mathbf{0}$, the projection from world to screen coordinates can be expressed as

$$\tilde{\mathbf{x}}_\mathbf{0} \sim \tilde{\mathbf{K}}_\mathbf{0}\mathbf{E}_\mathbf{0}\mathbf{p} = \tilde{\mathbf{P}}_\mathbf{0}\mathbf{p}$$

By knowing the z information or disparity value d_0, the 3D point p can be calculated

$$\mathbf{p} \sim \mathbf{E}_\mathbf{0}{}^{-1}\tilde{\mathbf{K}}_\mathbf{0}^{-1}\tilde{\mathbf{x}}_\mathbf{0}$$

Thus, this 3D point can be projected onto the other image plane

$$\tilde{\mathbf{x}}_\mathbf{1} \sim \tilde{\mathbf{K}}_\mathbf{1}\mathbf{E}_\mathbf{1}\mathbf{p} = \tilde{\mathbf{K}}_\mathbf{1}\mathbf{E}_\mathbf{1}\mathbf{E}_\mathbf{0}{}^{-1}\tilde{\mathbf{K}}_\mathbf{0}^{-1}\tilde{\mathbf{x}}_\mathbf{0} = \tilde{\mathbf{P}}_\mathbf{1}\tilde{\mathbf{P}}_\mathbf{0}^{-1}\tilde{\mathbf{x}}_\mathbf{0} = \mathbf{M}_\mathbf{10}\tilde{\mathbf{x}}_\mathbf{0}$$

Usually, the depth coordinates of pixels is not available. Although, for a planar scene, the last row of $\mathbf{P}_\mathbf{0}$ can be substituted with a plane equation, $\hat{\mathbf{n}}_\mathbf{0}.\mathbf{p}+c_0$. It maps points on the plane to $d_0 = 0$ values (Figure 2.3). Therefore, we can disregard the last column of $\mathbf{M}_\mathbf{10}$ and also its last row, as we set $d_0 = 0$, i.e., we do not care about the depth (z information) [42]. Hence, the mapping equation can be written as

$$\tilde{\mathbf{x}}_\mathbf{1} \sim \tilde{\mathbf{H}}_\mathbf{10}\tilde{\mathbf{x}}_\mathbf{0}$$

where $\tilde{\mathbf{x}}_\mathbf{1}$ and $\tilde{\mathbf{x}}_\mathbf{0}$ are 2D homogeneous coordinates and $\tilde{\mathbf{H}}_\mathbf{10}$ is a 3×3 homography matrix.

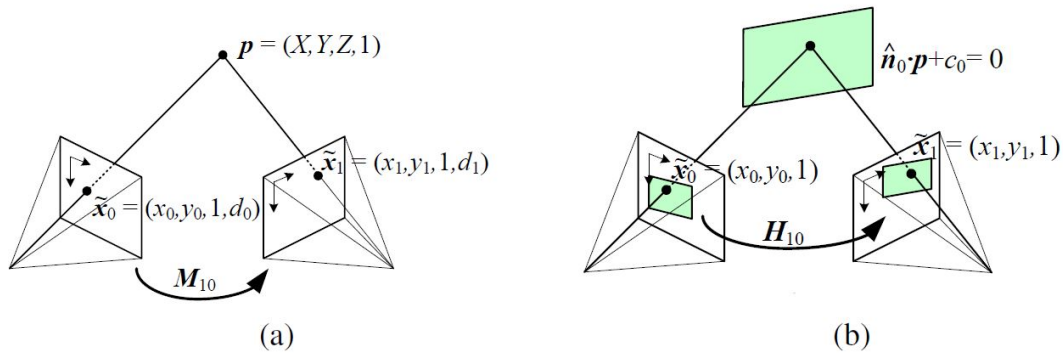

Figure 2.3: (a) Non-planar homography. (b) Planar homography [42].

2.6 Feature-based Alignment

Computation of 2D and 3D transformations that map features in one image to another is called **image registration**. One special case of this problem is **pose estimation**, which calculates the camera's position relative to a known scene or 3D object [42].

The problem of determining the motion between two or more sets of matched 2D or 3D points is known as **feature-based alignment**.

2.6.1 2D alignment using least squares

Let $(\mathbf{x_i}, \mathbf{x_i}')$ be a set of matched feature points. The relationship between these matched features in the form of planar transformation can be expressed as

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}; \mathbf{p})$$

We can calculate the motion parameters $\mathbf{p}$ using least squares by minimizing the sum of squared residuals

$$E_{LS} = \sum_i \parallel r_i{}^2 \parallel = \sum_i \parallel \mathbf{f}(\mathbf{x_i}; \mathbf{p}) - \mathbf{x_i}' \parallel$$

where

$$r_i = \mathbf{f}(\mathbf{x_i}; \mathbf{p}) - \mathbf{x_i}' = \hat{\mathbf{x}}_\mathbf{i}' - \tilde{\mathbf{x}}_\mathbf{i}'$$

is the difference between the measured location $\hat{\mathbf{x}}_\mathbf{i}'$ and its corresponding current predicted location $\tilde{\mathbf{x}}_\mathbf{i}' = \mathbf{f}(\mathbf{x_i}; \mathbf{p})$.

Motion models such as translation, similarity and affine have a linear relationship between the unknown parameters $\mathbf{p}$ the amount of motion $\Delta\mathbf{x} = \mathbf{x}' - \mathbf{x}$,

$$\Delta\mathbf{x} = \mathbf{x}' - \mathbf{x} = \mathbf{J}(\mathbf{x})\mathbf{p}$$

where $\mathbf{J} = \partial\mathbf{f}/\partial\mathbf{p}$ is the Jacobian of the transformation $\mathbf{f}$ with respect to the motion parameters $\mathbf{p}$. To solve this problem linear least squares (linear regression) can be applied

$$\begin{aligned} E_{LLS} &= \sum_i \parallel \mathbf{J}(\mathbf{x_i})\mathbf{p} - \Delta\mathbf{x} \parallel^2 \\ &= \mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x_i})\mathbf{J}(\mathbf{x_i})]\mathbf{p} - 2\mathbf{p}^T[\sum_i \mathbf{J}^T(\mathbf{x_i})\Delta\mathbf{x_i}] + \sum_i \parallel \Delta\mathbf{x_i} \parallel^2 \\ &= \mathbf{p}^T\mathbf{A}\mathbf{p} - 2\mathbf{p}^T\mathbf{b} + c \end{aligned}$$

To find the minimum we can solve the symmetric positive definite (SPD) system of normal equations

$$\mathbf{Ap} = \mathbf{b}$$

where

$$\mathbf{A} = \sum_i \mathbf{J}^T(\mathbf{x_i})\mathbf{J}(\mathbf{x_i})$$

is known as the Hessian and $\mathbf{b} = \sum_i \mathbf{J}^T(\mathbf{x_i})\Delta\mathbf{x_i}$ [42].

2.6.2 Iterative algorithms

Another approach to determine the feature-based alignment is to apply non-linear methods as most problems do not have a linear relationship between the measurements and the unknowns parameters. These problems are called **non-linear least squares** or **non-linear regression**.

2.6.3 RANSAC

In the case of having outliers among measurements, more robust least squares are required.

RANdom SAmple Consensus (RANSAC), method begins by choosing (at random) a subset of k correspondences to compute an initial guess for $\mathbf{p}$. The residuals of the full set of correspondences are calculated as

$$\mathbf{r_i} = \tilde{\mathbf{x}}'_\mathbf{i}(\mathbf{x_i}; \mathbf{p}) - \hat{\mathbf{x}}'_\mathbf{i}$$

where $\hat{\mathbf{x}}'_\mathbf{i}$ are the detected feature point locations and $\tilde{\mathbf{x}}_\mathbf{i}$ are the mapped locations.

The next step is to count the number of inliers that are within ϵ of their predicted location ($\| \mathbf{r_i} \| \leq \epsilon$). The random selection process is repeated S times and the sample

set with the largest number of inliers (with the smallest residual) is considered as the final solution. Then, the initial estimate of **p** or the full set of computed inliers are passed to the next steps [42].

2.6.4 Pose estimation

One special case of feature-based alignment is **pose estimation**, which calculates the camera's position relative to a known scene or 3D object from a set of 2D points, which is also called **extrinsic calibration**. The problem of estimating pose from three correspondences, which is the minimal amount of information necessary, is known as the **perspective-3-point-problem (P3P)**. It extends to larger numbers of points that is regarded as **PnP**.

Linear algorithms

The most straightforward way to retrieve the pose of the camera is to write down a set of linear equations

$$x_i = \frac{p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$

$$y_i = \frac{p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}$$

where (X_i, Y_i, Z_i) are the known 3D feature locations and (x_i, y_i) are the measured 2D features; see figure 2.4. These equations can be solved linearly in which the unknowns are the camera matrix **P**. In this way, both sides of the equation need to be multiplied by denominator. This algorithm is known as the direct linear transform

(**DLT**). To recover the 12 or 11 unknowns in $\mathbf{P}$, at least six correspondences between 2D and 3D locations are required.

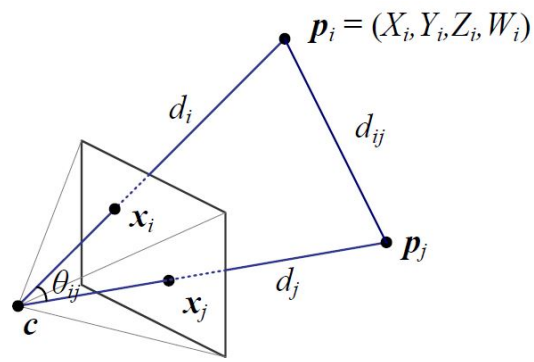

Figure 2.4: Pose estimation [42].

To recover the estimate unknowns in $\mathbf{P}$ more accurately, we can minimize the set of equations using non-linear least squares with a fewer iterations.

After the unknowns in $\mathbf{P}$ have been retrieved, we can calculate the intrinsic calibration matrix $\mathbf{K}$ and the rigid transformation $(\mathbf{R}, \mathbf{t})$ from the equation $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ by the way of QR factorization. When the camera is already calibrated, i.e., the matrix $\mathbf{K}$ is known, we can recover pose estimation using only three points (P3P).

2.7 Triangulation

The problem of recovering a point's 3D position from a set of corresponding image locations and known camera positions is called triangulation. This problem is the converse of the pose estimation [42].

One of the most straightforward ways to tackle this problem is to find the 3D point $\mathbf{p}$ that stands nearesto to all of the 3D rays corresponding to the 2D matching feature locations $\mathbf{x_j}$ detected by cameras $\mathbf{P_j} = \mathbf{K_j}[\mathbf{R_j}|\mathbf{t_j}]$, where $\mathbf{t_j} = -\mathbf{R_j}c_j$ and c_j

is the jth camera center; see figure 2.5. These rays begin at c_j in a direction $\hat{\nu_j} = \mathcal{N}(\mathbf{R_j}^{-1}\mathbf{K_j}^{-1}\mathbf{x_j})$. The closest point to $\mathbf{p}$ on this ray, which can be expressed as $\mathbf{q_j}$, minimizes the distance

$$\| c_j + d_j\hat{\nu}_j - \mathbf{p} \|^2$$

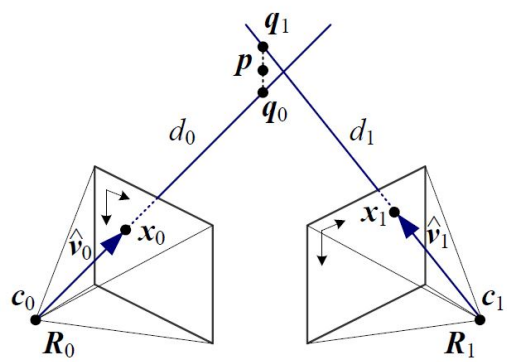

Figure 2.5: 3D point triangulation [42].

which has a minimum at $d_j = \hat{\nu}_j.(\mathbf{p} - c_j)$. Thus,

$$\mathbf{q_j} = c_j + (\hat{\nu}_j\hat{\nu}_j^T)(\mathbf{p} - c_j)$$

The squared distance between $\mathbf{p}$ and $\mathbf{q_j}$ is

$$r_j^2 = \| (\mathbf{I} - \hat{\nu}_j\hat{\nu}_j{}^T)(\mathbf{p} - c_j) \|^2$$

The optimal value for $\mathbf{p}$, which is the closest to all of the rays, can be retrieved using least squares fashion by summing over all the r_j^2 and obtaining the optimal value of $\mathbf{p}$,

$$\mathbf{p} = [\sum_j(\mathbf{I} - \hat{\nu}_j\hat{\nu}_j^T)]^{-1}[\sum_j(\mathbf{I} - \hat{\nu}_j\hat{\nu}_j^T)c_j]$$

Another way to solve this problem is to minimize the residual in the measurement equations

$$x_j = \frac{p_{00}^{(j)}X + p_{01}^{(j)}Y + p_{02}^{(j)}Z + p_{03}^{(j)}W}{p_{20}^{(j)}X + p_{21}^{(j)}Y + p_{22}^{(j)}Z + p_{23}^{(j)}W}$$

$$y_i = \frac{p_{10}^{(j)}X + p_{11}^{(j)}Y + p_{12}^{(j)}Z + p_{13}^{(j)}W}{p_{20}^{(j)}X + p_{21}^{(j)}Y + p_{22}^{(j)}Z + p_{23}^{(j)}W}$$

where $\{p_{00}^{(J)}...p_{23}^{(J)}\}$ are the known entries in camera matrix $\mathbf{P_j}$ and (x_j, y_j) are the measured 2D feature locations.

Homogeneous coordinates can be used, i.e., $\mathbf{p} = (X, Y, Z, W)$. Hence, the set of equations is homogeneous and can be solved as a singular value decomposition (SVD) or eigenvalue problem, i.e., looking for the smallest singular vector or eigenvector [42].

While working with triangulation, we should consider the issue of **chirality**, i.e., we should make sure that the reconstructed points are in front of all the cameras.

2.8 Structure from Motion, Epipolar Constraint

Structure from Motion (SFM) is the process of estimating the 3D structure and pose from image correspondences simultaneously.

Figure 2.6 illustrates a 3D point $\mathbf{p}$ being observed from two cameras whose relative position is known by a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. we can consider the first camera at the origin $c_0 = 0$ and at a the orientation of $\mathbf{R}_0 = \mathbf{I}$, without loss of generality. The vectors $\mathbf{t} = c_1 - c_0$, $\mathbf{p} - c_0$ and $\mathbf{p} - c_1$ are co-planar and define the basic of **epipolar constraint**, which can be written in terms of the measurements $\mathbf{x_0}$ and $\mathbf{x_1}$.

The observed point $\mathbf{p}$ in the first image, $\mathbf{p}_0 = d_0\hat{\mathbf{x_0}}$ is mapped into the second

image by the transformation

$$d_1\hat{\mathbf{x}}_\mathbf{1} = \mathbf{p} = \mathbf{R}\mathbf{p_0} + \mathbf{t} = \mathbf{R}(d_0\hat{\mathbf{x}}_\mathbf{0}) + \mathbf{t}$$

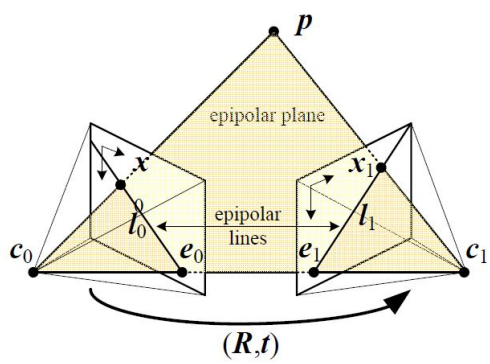

Figure 2.6: Epipolar geometry [42].

where $\hat{\mathbf{x}}_\mathbf{j} = \mathbf{K_j}^{-1}\mathbf{x_j}$ are the ray direction vectors. By multiplying both sides with $\mathbf{t}$ we have

$$d_1[\mathbf{t}]_\times\hat{\mathbf{x}}_\mathbf{1} = d_0[\mathbf{t}]_\times\mathbf{R}\hat{\mathbf{x}}_\mathbf{0}$$

Taking the dot product of both sides with $\hat{\mathbf{x}}_\mathbf{1}$ we have

$$d_0\hat{\mathbf{x}}_\mathbf{1}^T([\mathbf{t}]_\times\mathbf{R})\hat{\mathbf{x}}_\mathbf{0} = d_1\hat{\mathbf{x}}_\mathbf{1}^T[\mathbf{t}]_\times\hat{\mathbf{x}}_\mathbf{1}$$

Hence, the basic epipolar constraint can be expressed as

$$\hat{\mathbf{x}}_\mathbf{1}^T\mathbf{E}\hat{\mathbf{x}}_\mathbf{0}^T = 0$$

where

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$$

is known as the **essential matrix**.

To recover the matrix $\mathbf{E}$ we need N corresponding measurements $\{(x_{i0}, x_{i1})\}$. Then, N homogeneous equations in the nine elements of $\mathbf{E} = \{e_{00}...e_{22}\}$ can be written as,

$$\begin{array}{ccccccc} x_{i0}x_{i1}e_{00} & + & y_{i0}x_{i1}e_{01} & + & x_{i1}e_{02} & + & \\ x_{i0}y_{i1}e_{00} & + & y_{i0}y_{i1}e_{11} & + & y_{i1}e_{12} & + & = 0 \\ x_{i0}e_{20} & + & y_{i0}e_{21} & + & e_{22} & & \end{array}$$

where $\mathbf{x_{ij}} = (x_{ij}, y_{ij}, 1)$. This can be denoted as

$$[\mathbf{x_{i1}}\mathbf{x_{i0}}^T] \otimes \mathbf{E} = \mathbf{Z_i} \otimes \mathbf{E} = z_i.\mathbf{f} = 0$$

where $\otimes$ shows an element-wise multiplication and summation of matrix elements, and z_i and $\mathbf{f}$ are the rasterized (vector) forms of the $\mathbf{Z_i} = \hat{\mathbf{x}}_{\mathbf{i1}}\hat{\mathbf{x}}_{\mathbf{i0}}^T$ and $\mathbf{E}$ matrices. Having $N \leq 8$ equations, the unknowns in $\mathbf{E}$ can be calculated by applying an SVD.

In order to recover the direction $\hat{\mathbf{t}}$, we take advantage of the fact that essential matrix $\mathbf{E}$ is singular, i.e., $\hat{\mathbf{t}}^T\mathbf{E} = 0$. Performing an SVD of $\mathbf{E}$ we have

$$\mathbf{E} = [\hat{\mathbf{t}}]_\times \mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \begin{bmatrix} \mathbf{u_0} & \mathbf{u_1} & \hat{\mathbf{t}} \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v_0}^T \\ \mathbf{v_1}^T \\ \mathbf{v_2}^T \end{bmatrix}$$

As $\mathbf{E}$ is computed from noisy measurements, $\hat{\mathbf{t}}$ is obtained from the singular vector

related with the smallest singular [42].

The next step is to estimate the rotation matrix $\mathbf{R}$. The cross-product operator $\hat{\mathbf{t}}$ projects a vector onto a set of orthogonal basis vectors that contains $\hat{\mathbf{t}}$, which sets the $\hat{\mathbf{t}}$ component zero, and rotates the other two by 90°,

$$[\hat{\mathbf{t}}]_\times = \mathbf{SZR}_{90^\circ}\mathbf{S}^T = \begin{bmatrix} s_0 & s_1 & \hat{\mathbf{t}} \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s_0}^T \\ \mathbf{s_1}^T \\ \hat{\mathbf{t}}^T \end{bmatrix}$$

where $\hat{\mathbf{t}} = s_0 \times s_1^t$. Therefore we obtain

$$\mathbf{E} = [\hat{\mathbf{t}}]_\times \mathbf{R} = \mathbf{SZR}_{90^\circ}\mathbf{S}^T\mathbf{R} = \mathbf{U\Sigma V}^T$$

Which shows that $\mathbf{S} = \mathbf{U}$. In the case of noise free essential matrix, $\mathbf{\Sigma} = \mathbf{Z}$. Thus we have

$$\mathbf{R}_{90^\circ}\mathbf{U}^T\mathbf{R} = \mathbf{V}^T$$

and

$$\mathbf{R} = \mathbf{UR}_{90^\circ}\mathbf{V}^T$$

Here we have a problem. Both $\mathbf{E}$ and $\hat{\mathbf{t}}$ are known up to a sign. Moreover, if we flip the signs of the matrices $\mathbf{U}$ and $\mathbf{V}$ we still can have a valid SVD. Hence, all four possible rotation matrices need to be considered

$$\mathbf{R} = \pm\mathbf{UR}_{\pm 90^\circ}\mathbf{V}^T$$

and retain the two whose determinant $|\mathbf{R}| = 1$. Afterward, both possible signs for the translation direction $\pm\hat{\mathbf{t}}$ are considered in combination with the potential rotations matrices. The matrices whose the largest number of points are seen in front of both cameras are selected.

The process described above is known as the normalized **eight-point algorithm**, which is a way to estimate the camera motion from correspondences, but it is not the only method.

Bundle Adjustment

The most accurate way to retrieve structure and motion is the robust non-linear minimization of the measurement (re-projection) errors, which is called bundle adjustment.

The main difference between iterative pose estimation and full bundle adjustment is that the feature location measurements $\mathbf{x_{ij}}$ rely not only on the point (track index) i but also on the camera pose index j,

$$\mathbf{x_{ij}} = \mathbf{f}(\mathbf{p_i}, \mathbf{R_j}, c_j, \mathbf{K_j})$$

Furthermore, the 3D point positions $\mathbf{p_i}$ are being updated simultaneously [42].

Chapter 3

Visual Odometry

3.1 Introduction

Odometry is the process of estimating the position and orientation of an agent (e.g., vehicle, human, and robot) relative to its initial location. For autonomous navigation, obstacle detection and avoidance, motion tracking and many other application an agent must know its position over time. Vision-based odometry or **Visual Odometry (VO)** is a technique used to estimate the ego motion of an agent using only the input of a single or multiple cameras attached to it [35].

Compared to other sensor-based localization systems, cameras are more reliable and robust [33]. As the autonomous vehicles and Advanced Driver Assistant systems (ADAS) are among the hot topics these days, detection of road edges, lanes, pedestrian, other vehicles and their transitions are crucial for these systems. The images captured by a camera can provide a large amount of meaningful information to be used for these and many other purposes. Therefore, utilizing camera is beneficial. Furthermore, compared with other localization sensors, optical cameras are cheaper.

Indeed, they are passive, so they do not have the problem of interference with other active sensors like LIDAR or RADAR. Cameras are also small, lightweight, low powered and flexible (they can be employed in any vehicle e.g. air, land, underwater) [1].

3.1.1 VO challenges

A static scene with enough texture and good lighting are essential for VO to work efficiently. Moreover, neighboring frames should have sufficient scene overlap [35]. Therefore, many factors such as low-textured background scene, direct sunlight or shadows, non flat terrains make the VO challenging in outdoor environment. Mainly, the challenges in VO are computational cost and image condition [35].

In particular, monocular Visual Odometry suffers from scale ambiguity. It is because of the fact that 3D information can not be derived from only one camera. Estimation of the scale become challenging when surface of the road is not flat and uneven, or the road slope changes frequently, which may lead to inaccurate estimation of trajectory.

3.1.2 VO applications

Visual Odometry is used in various applications varying from robotics to augmented reality. VO is mainly employed for navigation and to reach targets efficiently and at the same time to avoid obstructions in its path. Another application of VO is in the autonomous take-off and landing of unmanned aerial vehicles (UAVs).

Indeed, in the automotive industry, VO plays a big role. The dream of a fully autonomous vehicle is still ongoing and is a hot topic in academia as well as in

industry, which adds to the VO's importance. VO is used in ADAS systems, such as vision-based assisted braking systems, vehicle localization, obstacle detection and many other purposes. In cases where the GPS signal is lost or more accuracy is needed, VO is employed. Also, compared to LIDAR systems it is more economical.

3.1.3 Stereo vs. Monocular VO

Stereo and monocular camera systems are used widely today for various applications. Both give a continuous visual images, which can later be used for any particular use.

Stereo camera usually consists of a two or more camera system rigidly fixed to a platform in a known geometry. Visual Odometry estimation using such sensors is called stereo Visual Odometry. The distance between the two lenses of the cameras (the stereo baseline) is fixed and known, thus the scale parameters can be recovered. In other words, the 3D information can be extracted in this scenario, which adds to the information available. On the other hand, binocular cameras are more expensive and their calibration is a bottleneck than the monocular cameras; keeping a single calibrated camera is much easier than maintaining a calibrated constant baseline between a pair of cameras [16]. While the costs of every day use cameras have considerably decreased, cameras for VO application are not cheap as they must be resistant to severe temperatures, weather and jitters, and support high frame rates [40].

Monocular cameras are single camera setups and are used in monocular Visual Odometry. Using a monocular camera diminish the effect of calibration errors in motion estimation. The main motivations for using the monocular cameras are the low

cost and the easy set up in a variety of applications, such as cellular phones. Nevertheless, monocular vision systems suffer from scale uncertainty. In these scenarios other information and assumptions are needed. For example, it is assumed that the surface of the road is flat, but when the surface is uneven or the road slope changes dramatically, the image scaling factor will become difficult to estimate resulting error in the estimation of trajectory. Furthermore, synchronization and calibration are more demanding with stereo cameras than with monocular cameras [35].

3.2 Formulation of the VO Problem

Consider a moving car. The cameras mounted on this car are taking images at discrete time instants k. For a monocular system, the set of images taken at times k is indicated by $I_{0:n} = \{I_0, ..., I_n\}$. There are a left and a right image in case of a stereo system, which can be denoted by $I_{l,0:n} = \{I_{l,0}, ..., I_{l,n}\}$ and $I_{r,0:n} = \{I_{r,0}, ..., I_{r,n}\}$. Figure 3.1 illustrates this setting.

For simplicity, we can assume that the camera coordinate frame is the moving agent's coordinate frame. For a stereo system, the coordinate system of the left camera can be considered as the origin.

Let $\mathbf{T}_{k,k-1} \in \mathbb{R}^{4\times4}$ be the **rigid body transformation** that relates camera positions at neighboring time instants $k-1$ and k

$$\mathbf{T}_{k,k-1} = \begin{bmatrix} \mathbf{R}_{k,k-1} & \mathbf{t}_{k,k-1} \\ \mathbf{0} & 1 \end{bmatrix}$$

where $\mathbf{R}_{k,k-1}$ is the rotation matrix, and $\mathbf{t}_{k,k-1} \in \mathbb{R}^{3\times1}$ the translation vector. The set $\mathbf{T}_{1:n} = \{\mathbf{T}_{1,0}, ..., \mathbf{T}_{n,n-1}\}$ includes all succeeding motions. For simplicity, $\mathbf{T}_k$

can be written equivalently as $\mathbf{T}_{k,k-1}$. Lastly, let $\mathbf{C}_{0:n} = \{\mathbf{C}_0, ..., \mathbf{C}_n\}$ be the set of camera poses (full trajectory of the camera) with respect to the initial coordinate frame at time $k = 0$. The present pose $\mathbf{C}_n$ can be calculated by concatenating all the transformations $\mathbf{T}_k (k = 1, ..., n)$. Thus,

$$\mathbf{C}_n = \mathbf{C}_{n-1}\mathbf{T}_n$$

where $\mathbf{C}_0$ is the camera pose at time $k = 0$.

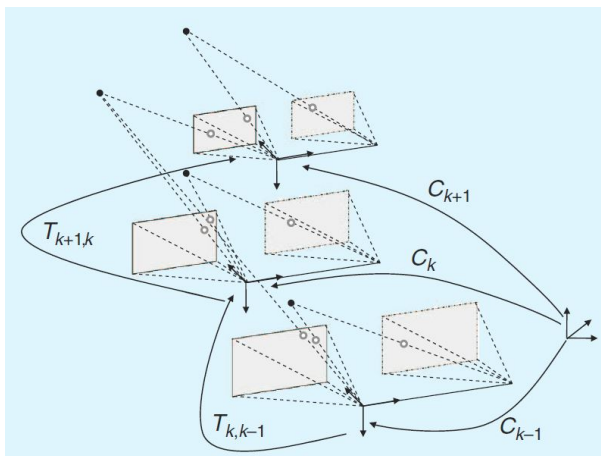

Figure 3.1: An illustration of the Visual Odometry problem [35].

Figure 3.2 summarizes the VO pipeline. Firstly, 2-D features are detected and matched in each new image I_k (or image pair in the stereo scenario), with the previous images. **Image correspondences** are the features that are the reprojection of the same 3-D point in different images. Then, the relative motion $\mathbf{T}_k$ is calculated between the time $k-1$ and k. There exists three different approaches to compute the $\mathbf{T}_k$ matrix, which depends on the dimension of image correspondences. The last step consists of an iterative improvement to acquire a more precise estimate of the trajectory.

Figure 3.2: A typical Visual Odometry pipeline.

3.2.1 Camera Modeling and Calibration

There are various camera models to select from when dealing with Visual Odometry, e.g. catadioptric, spherical, omni-directional and perspective projection. In this section, we only explain perspective model as we use it in the algorithm.

Perspective Camera Model

The pinhole projection system is the most common model for perspective cameras. In this model the image formation is described as the intersection of the light rays from the objects towards the projection center (center of the lens) [35]. See figure 3.3 for more details. Assume that $\mathbf{X} = [x, y, z]^T$ be a 3-D scene point and $\mathbf{p} = [u, v]^T$ its projection on the image plane evaluated in pixels. **Perspective projection** equation can describe the mapping from the 3-D world to the 2-D image plane:

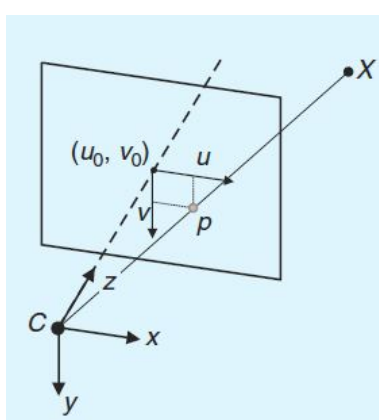

Figure 3.3: Perspective camera model [35].

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{KX} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

'where λ is the depth factor, f_u and f_v the focal lengths, and u_0, v_0 the image coordinates of the projection center. These parameters are called intrinsic parameters.' Consider $\tilde{\mathbf{p}} = [\tilde{u}, \tilde{v}, 1]^T = \mathbf{K}^{-1}[u, v, 1]^T$ as the normalized image coordinates. For the sake of simplicity, this notation will be used.

3.3 Motion Estimation

This step, which is the main computation step for any VO system, is where the actual motion is estimated. Image correspondences are the input to this step. It deals with the computation of the camera motion, $\mathbf{T}_k$, between the current image and the previous image (I_{k-1} and I_k) at time instants $k-1$ and k, respectively. There are three different methods for computing $\mathbf{T}_k$ matrix [35].

- 2D-to-2D: both f_{k-1} and f_k are described in 2D image coordinates.
- 3D-to-3D: both f_{k-1} and f_k are defined in 3D.
- 3D-to-2D: f_{k-1} are identified in 3D and f_k are their corresponding 2D reprojections on the image I_k.

3.3.1 2D-to-2D

Estimating the Essential Matrix

Essential matrix, $\mathbf{E}$, described the geometric relations between two images I_k and I_{k-1} of a calibrated camera, which can be written as

$$\mathbf{E}_k \simeq \hat{\mathbf{t}_k}\mathbf{R}_k$$

where $\mathbf{t}_k = [t_x, t_y, t_z]^T$ and

$$\hat{\mathbf{t}}_\mathbf{k} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

The symbol $\simeq$ indicates that the equality is valid up to a scale.

The essential matrix can be recovered from 2D-to-2D feature correspondences using epipolar constraint, and rotation and translation can be computed from $\mathbf{E}$. As discussed in section 2.8, this constraint can be written as $\tilde{\mathbf{p}'}\mathbf{E}\tilde{\mathbf{p}} = 0$, where $\tilde{\mathbf{p}'}$ is a feature location in one image (I_k) and $\tilde{\mathbf{p}}$ is the location of its corresponding feature in another image (I_{k-1}).

The minimal case solution needs five 2D-to-2D correspondences and an effective implementation can be found in [31]. A simple approach for $n \geq 8$ noncoplanar points is the Longuet-Higgins eight-point algorithm [23]. In this algorithm each feature match gives a constraint as follow

$$\begin{bmatrix} \tilde{u}\tilde{u'} & \tilde{u'}\tilde{v} & \tilde{u'} & \tilde{u}\tilde{v'} & \tilde{v}\tilde{v'} & \tilde{v'} & \tilde{u} & \tilde{v} & 1 \end{bmatrix} \mathbf{E} = 0$$

where $\mathbf{E} = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{bmatrix}^T$.

Therefore, we have a linear equation system $\mathbf{AE} = 0$ in which the parameters of $\mathbf{E}$ can be recovered using singular value decomposition (SVD) [23]. Having more than eight points leads to an overdetermined system to solve in the least squares sense and provides a degree of robustness to noise.

As discussed in section 2.8, the SVD of $\mathbf{A}$ can be written as $\mathbf{A} = \mathbf{USV}^T$, and the estimate of $\mathbf{E}$ with $\| \mathbf{E} \| = 1$ can be obtained as the last column of $\mathbf{V}$. A correct essential matrix after SVD is $\mathbf{E} = \mathbf{USV}^T$ and has $\text{diag}(\mathbf{S}) = s, s, 0$. Then $\mathbf{E}$ needs to be projected onto the space of valid essential matrices in order to satisfy the constraints. The projected essential matrix is $\bar{\mathbf{E}} = \mathbf{U}\ \text{diag}\{1, 1, 0\}\mathbf{V}^T$.

Extracting R and t from E

Based on the discussion in section 2.8, the rotation and translation matrices can be obtained from $\bar{\mathbf{E}}$ in which there are four distinct solutions for $\mathbf{R}$, $\mathbf{t}$ for one essential matrix. The four solutions are

$$\mathbf{R} = \mathbf{U}(\pm\mathbf{W}^T)\mathbf{V}^T$$

$$\tilde{\mathbf{t}} = \mathbf{U}(\pm\mathbf{W})\mathbf{SU}^T$$

where,

$$\mathbf{W}^T = \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Next, by applying triangulation of a point and the fact that point lies in front of

both cameras, the correct answer is obtained.

Computing the Relative Scale

The translation matrix that is estimated in the previous section is valid up to scale. Hence, the relative scale is required. One solution is to triangulate 3D points $\mathbf{X}_{k-1}$ and $\mathbf{X}_k$ from two neighboring image pairs. From the corresponding 3D points, the relative distances between the two 3D points can be obtained. Therefore, the distance ratio r between $\mathbf{X}_{k-1}$ and $\mathbf{X}_k$ is expressed as

$$r = \frac{\| \mathbf{X}_{k-1,i} - \mathbf{X}_{k-1,j} \|}{\| \mathbf{X}_{k,i} - \mathbf{X}_{k,j} \|}$$

To have more robust estimate of scale, the scale ratios for many point pairs are calculated and the mean is taken. Then, the translation vector $\mathbf{t}$ is scaled with r. This computation needs features to be matched or tracked over at least three frames. An alternative way is to use the trifocal constraint between three-view matches of 2D features [14].

Visual Odometry with 2D-to-2D motion estimation algorithm can be written as the following steps [35]

VO from 2D-to-2D correspondences algorithm

- 1) Get new frame I_k
- 2) Obtain and match features between I_{k-1} and I_k
- 3) Calculate essential matrix for image pair I_{k-1}, I_k
- 4) Compute $\mathbf{R}_k$ and $\mathbf{t}_k$ from essential matrix, and form $\mathbf{T}_k$

- 5) Calculate relative scale and rescale $\mathbf{t}_k$ accordingly
- 6) Concatenate transformation, $\mathbf{C}_k = \mathbf{C}_{k-1}\mathbf{T}_k$
- 7) Repeat from 1).

3.3.2 3D-to-3D

In this scenario, the camera motion $\mathbf{T}_k$ can be recovered by calculating the transformation between two sets of 3D feature, which is available in stereo VO.

To solve this problem, the $\mathbf{T}_k$ matrix that minimizes the L_2 distance between the two 3-D feature sets need to be found

$$\underset{\mathbf{T}_k}{\operatorname{argmin}} \sum_i \parallel \tilde{\mathbf{X}}^i_{\mathbf{k}} - \mathbf{T}_k \tilde{\mathbf{X}}^i_{\mathbf{k-1}} \parallel$$

where the superscript indicates the ith feature, and $\tilde{\mathbf{X}}_{\mathbf{k-1}}$, $\tilde{\mathbf{X}}_{\mathbf{k}}$ represent the homogeneous coordinates of the 3D points, i.e., $\tilde{\mathbf{X}} = [x, y, z, 1]^T$.

The minimum requirement is three 3D-to-3D correspondences. For the case of $n \geq 3$ correspondences, one possible method can be found in [2]. In this method the translation part is calculated as the difference of the centroids of the 3D feature sets and the rotation part using singular value decomposition. Therefore, translation can be written as

$$\mathbf{t}_k = \overline{\mathbf{X}}_k - \mathbf{R}\overline{\mathbf{X}}_{k-1}$$

where over line denotes the arithmetic mean value.

The rotation can be recovered as

$$\mathbf{R}_k = \mathbf{V}\mathbf{U}^T$$

where $\mathbf{U}\mathbf{S}\mathbf{V}^T = \text{svd}((\mathbf{X}_{k-1} - \overline{\mathbf{X}}_{k-1})(\mathbf{X}_k - \overline{\mathbf{X}}_k)^T)$ and $\mathbf{X}_{k-1}$ and $\mathbf{X}_k$ are sets of corresponding 3-D points.

The resulting transformation has absolute scale, therefore, scale estimation step is not needed here.

3.3.3 3D-to-2D

In [32] we can see that motion estimation from 3D-to-2D correspondences is more precise than from 3D-to-3D correspondences because it minimizes the image reprojection error instead of the 3D-to-3D feature position error.

In this method, the transformation $\mathbf{T}_k$ is recovered from stereo data or, in the case of monocular odometry, from triangulation of the image measurements $\mathbf{p}_{k-1}$ and $\mathbf{p}_{k-2}$, which needs image correspondences from three views.

The general solution is to obtain $\mathbf{T}_k$ that minimizes the image reprojection error

$$\underset{\mathbf{T}_k}{\text{argmin}} \sum_i \parallel \mathbf{p_k}^i - \tilde{\mathbf{p}}^i_{\mathbf{k-1}} \parallel^2$$

where $\tilde{\mathbf{p}}^i_{\mathbf{k-1}}$ is the reprojection of the 3D point $\mathbf{X_{k-1}}^i$ into image I_k according to the transformation $\mathbf{T}_k$. This problem is known as **perspective-from-n-points (PnP)**. Various solutions to this problem can be found in the literature [29]. The minimal case solution needs three 3D-to-2D correspondences (P3P) [7] and a fast implementation of P3P is described in [19].

A simple solution to the PnP problem for $n \geq 6$ points is the linear transformation algorithm [14]. One 3D-to-2D point correspondence gives two constraints for $\mathbf{P}_k = [\mathbf{R}|\mathbf{t}]$, which can be written as

$$\begin{bmatrix} 0000 & -x & -y & -z & -1 & x\tilde{v} & y\tilde{v} & z\tilde{v} & \tilde{v} \\ x & y & z & 1 & 0000 & -x\tilde{u} & -y\tilde{u} & -z\tilde{u} & -\tilde{u} \end{bmatrix} \begin{bmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{bmatrix} = 0$$

where x, y, z are the coordinates of the 3D points $\mathbf{X}_{k-1}$ and each $[p^j]^T$ is a four vector (the jth row of $\mathbf{P}_k$) [35]. Therefore, we have a linear system of equations as $\mathbf{AP} = 0$. The unknowns of $\mathbf{P}$ can be calculated using SVD. Then, the rotation and translation matrices are obtained from $\mathbf{P}_k = [\mathbf{R}|\mathbf{t}]$.

This method assumes that the 2D image points merely are extracted from one camera, i.e., for the case of a stereo camera, the 2D image points are from either the left or the right camera. For the case of monocular odometry, it is needed to triangulate 3D points and recover the pose from 3D-to-2D matches, which is referred to as SFM. In other words, beginning from two views, the initial set of 3D points and the first transformation are calculated from 2D-to-2D feature matches. Next transformations are recovered from 3D-to-2D feature correspondences. Therefore, it is necessary that features be matched or tracked over at least three frames. New 3D features are triangulated when a new transformation is calculated and added to the set of 3D features. The main challenge of this method is to retain a constant and precise set of triangulated 3D features and to generate 3D-to-2D feature matches for at least three subsequent frames [35].

It is better to use the 2D-to-2D and 3D-to-2D methods compared to the 3D-to-3D method for motion computation [32]. It is because of the fact that there is more ambiguity in the depth direction of the triangulated 3D points, which impacts the motion estimation process. In other words, in the 3D-to-3D case, the 3D position error is minimized but in the 3D-to-2D case the image reprojection error is minimized [35].

In the monocular case, as the 2D-to-2D method does not use triangulation it is better than the 3D-to-2D case. Although, in practice, the 3D-to-2D method is used more commonly because of its faster data association. As discussed before, the 2D-to-2D case needs a minimum of five point matches bur just three matches are required in the 3D-to-2D case, which helps the motion estimation process runs faster.

3.4 Implemented Algorithm

In this section, the pipeline of the monocular VO algorithm implemented in this thesis is explained.

The goal of this research is an efficient and robust Visual Odometry algorithm for estimating the ego motion of a ground vehicle which works in real time. The basic steps are as follow:

- Feature extraction
- Feature matching
- Motion estimation
- Filtering

Feature Extraction

In the feature extraction step, the input images are filtered with 5×5 blob and corner masks, as illustrates in Figure 3.4. Afterwards, on the resulting filtered images we apply non-maximum and non-minimum suppression [30]. Non-maximum (non-minimum) suppression can be formulated as local maximum (minimum) search, where a local maximum (minimum) is greater (smaller) than all its neighbors (excluding itself). Therefore, the feature that are obtained can be in one of four classes (corner min, corner max, blob max, blob min).

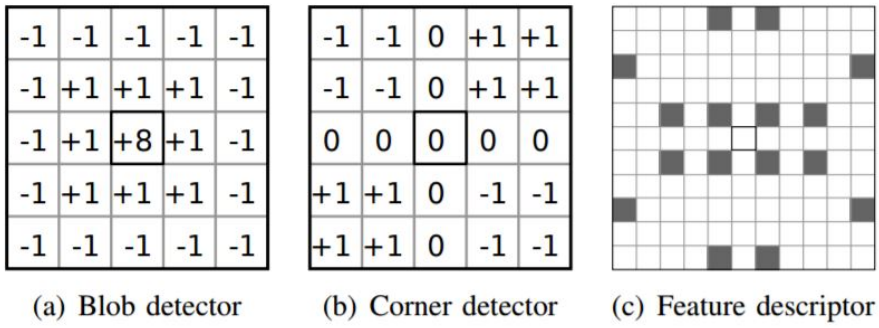

Figure 3.4: Blob and corner detector and feature descriptor [10].

Feature Matching

Consider two feature points, we apply 11×11 block windows of Sobel filter and compare their responses using the sum of absolute differences (SAD) error metric [10]. The sum of absolute differences measures the similarity between image blocks which is computed by taking the absolute difference between each pixel in the original block and the corresponding pixel in the block being used for comparison. Then, these differences are summed to show a simple metric of block similarity. To make the matching process faster, we quantize the Sobel responses to 8 bits and instead of

summing over the whole block window, we sum the differences over a sparse set of 16 locations, i.e., Figure 3.4(c).

In this step we also apply bucketing, which helps to reduced the number of features and spread features uniformly. In this case, the image is divided into many non-overlapping rectangles (bucket). In every bucket a maximal number of feature points are reserved. Hence, in the case that the scene contains several moving objects, not all features would be on moving objects.

Motion Estimation

In this section, we employ the 2D-to-2D motion estimation method, which is completely described in details in sections 3.3.1 and 2.8.

Given all feature correspondences from the previous steps, we compute the camera motion (essential matrix) using eight-point algorithm. Next, we project feature points from the previous frame into 3D via triangulation by having at least three subsequent frames. Therefore, we can apply the chirality constraint to disambiguate between the resulting rotation matrices that are obtained from decomposing the essential matrix via singular value decomposition (SVD).

For estimating the scale, we assume that the camera is moving at a known and fixed height from the ground. Further, it is considered that the ground in locally planar. Moreover, to be robust against outliers, we cover our estimation approach into a RANSAC method. All inliers of the winning iteration are then used for processing the parameters which gives the final transformation $(\mathbf{r}, \mathbf{t})$ [10].

Filtering

In order to refine the estimated parameters ($\mathbf{R}$ and $\mathbf{t}$) that are obtained from the previous steps, we run different filters on top of our Visual Odometry algorithm. This step as it needs a complete and detailed analysis will be discussed in the next chapter.

Chapter 4

Problem Formulation

This chapter contains the filtering approach that is divided into three parts. In each section the general algorithm of each filter has been presented, then the details of each filter that has been applied in the Visual Odometry system is demonstrated.

Firstly, the output of the Visual Odometry before filtering has been studied in details. We are interested in estimation of the position and orientation of the moving agent or equivalently the camera. Therefor, our sate consists of translation and rotation in X, Y and Z direction.

Here we look into the translation and rotation of the sequences 04, 06 and 08 of KITTI dataset, as the sequence 04 (figure 5.1a) is the simplest case without any turning and sequence 08 (figure 5.1e) is the case with a more complex path than sequence 06 (figure 5.1b).

Let t_x, t_y and t_z represent the translation in X, Y and Z axis direction in meter. Similarly, r_x, r_y and r_z show the orientation around X, Y and Z axis (yaw, pitch and roll) in radian.

The resulting estimate for sequences 04, 06, and 08 are presented in figures 4.1,

4.2, 4.3, 4.4, 4.5 and 4.6 respectively.

We can see that when a turn occurs there is a sudden jump in r_y. See figure 4.4. Continuing with dataset 08, we see the the sudden jumps in the results more noticeable than previous one, figure 4.6. Moreover, we can see that while turning the value for t_x is not zero, it is changing.

To make it more clear, imagine the car is moving straight with constant velocity. The only parameter that is changing is the t_z, which is showing the range information of the car. As the car turns, the value for t_x starts changing. t_y is the camera's height from the ground, which should be a constant but not when the road is not flat or the camera shakes. Furthermore, while the car turning the orientation around Y axis starts changing.

By looking at the translation and rotation estimates for the car moving in a straight line, we can see that the plots for translation and rotation are smooth and follows the constant velocity or constant acceleration model. Thus, we start with a simple Kalman filter with constant velocity and in the following the results before and after applying KF are presented.

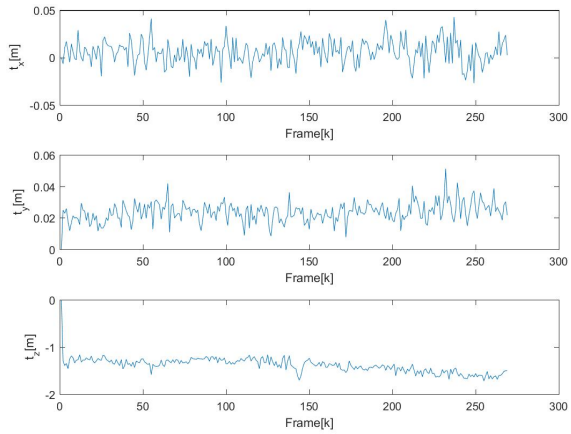

Figure 4.1: Translation before filtering for sequence 04

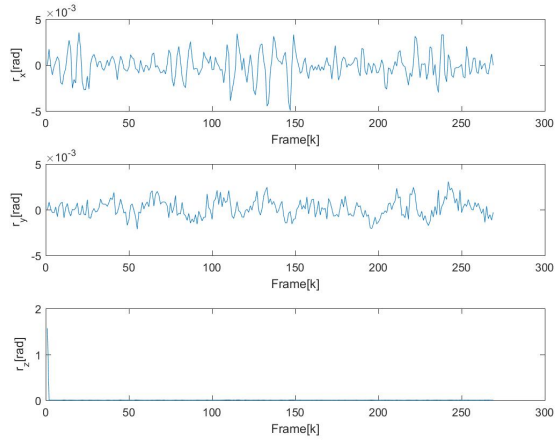

Figure 4.2: Rotation before filtering for sequence 04

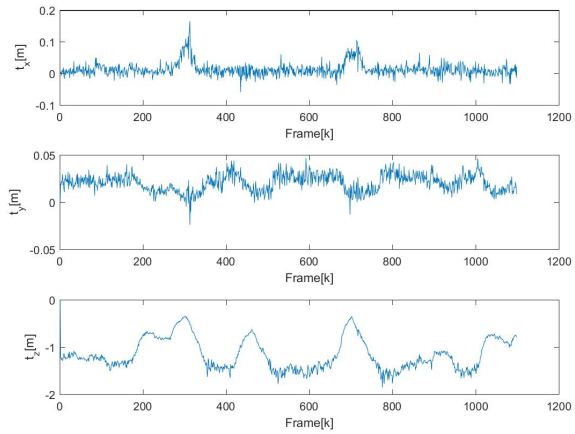

Figure 4.3: Translation before filtering for sequence 06

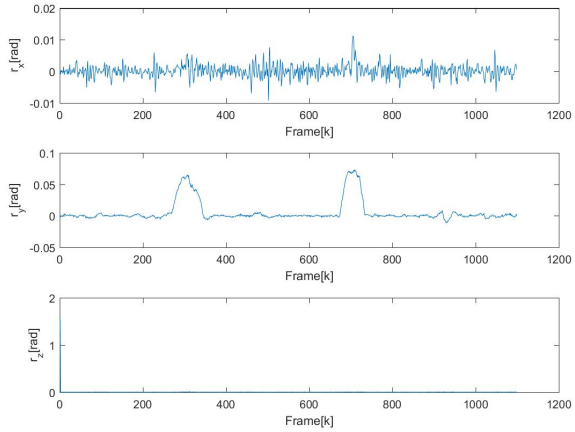

Figure 4.4: Rotation before filtering for sequence 06

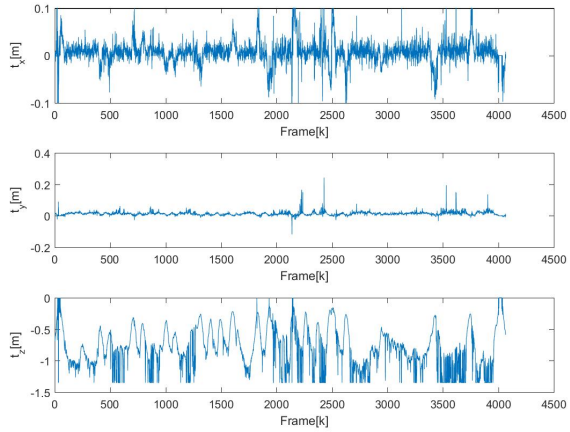

Figure 4.5: Translation before filtering for sequence 08

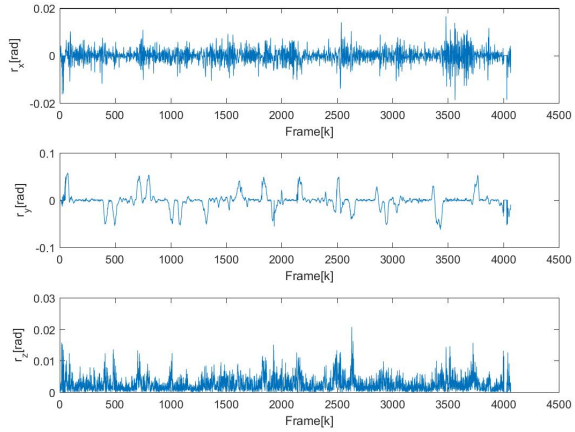

Figure 4.6: Rotation before filtering for sequence 08

4.1 Kalman Filter

This section contains the equations for the Kalman filter (KF). The Kalman filter is a state estimator in linear dynamic systems [3]. Kalman Filters are extremely useful for pose estimation of a linear system.

Assume a discrete-time linear dynamic system, which can be described by its plant equation

$$x(k+1) = F(k)x(k) + \nu(k),$$

where $x(k)$ is the n_x-dimensional state vector, $F(k)$ is the state transition matrix and $v(k)$, is the zero-mean white Gaussian process noise sequence with covariance $Q(k)$.

The measurement equation can be written as

$$z(k) = H(k)x(k) + w(k),$$

with $H(k)$ the observation matrix and $w(k)$ the zero-mean white Gaussian measurement noise sequence with covariance $R(k)$. The measurement and process noise are assumed to be independent.

Let $Z^k \triangleq \{z(i), i \leq k\}$ denote the sequence of observations available at time k. The conditional mean can be defined as $\hat{x}(j|k) \triangleq E[x(j)|Z^k]$.

The covariance associated with the estimate $x(j)$ given the data Z^k is

$$P(j|k) \triangleq E[[x(j) - \hat{x}(j|k)][x(j) - \hat{x}(j|k)]'|Z^k] = E[\tilde{x}(j|k)\tilde{x}(j|k)'|Z^k]$$

where the estimation error is $\tilde{x}(j|k) \triangleq x(j) - \hat{x}(j|k)$.

One cycle of the Kalman filter can be written as follows:

State prediction, $\hat{x}(k+1|k) = F(k)\hat{x}(k|k)$.

State prediction covariance, $P(k+1|k) = F(k)P(k|k)F(k)' + Q(k)$.

Measurement prediction, $\hat{z}(k+1|k) = H(k+1)\hat{x}(k+1|k)$.

Innovation covariance, $S(k+1) = H(k+1)P(k+1|k)H(k+1)' + R(k+1)$.

Filter gain, $W(k+1) \triangleq P(k+1|k)H(k+1)'S(k+1)^{-1}$.

The updated state estimate, $\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)\nu(k+1)$ where $\nu(k+1) \triangleq z(k+1) - \hat{z}(k+1|k) = \tilde{z}(k+1|k)$ is called the innovation or measurement residual.

Updated state covariance, $P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1)W(k+1)'$.

System details

On top of the estimation procedure of VO, we place a standard Kalman filter, assuming constant acceleration.

To this end, let the state vector containing the transformation parameters (the translation and orientation) be divided by the time between frames Δ_t which is written as,

$$\mathbf{v} = \begin{bmatrix} \mathbf{r} \\ \mathbf{t} \end{bmatrix} / \Delta_t = \begin{bmatrix} r_x & r_y & r_z & t_x & t_y & t_z \end{bmatrix}^T / \Delta_t$$

.

where $\mathbf{v}$ denotes velocity and $r_x, r_y, r_z, t_x, t_y, t_z$ contains the orientation and translation values with respect to 3D axis.

The plant equation is given by

$$x(k+1) = \begin{bmatrix} \mathbf{I} & \Delta_t \mathbf{I} \\ 0 & \mathbf{I} \end{bmatrix} x(k) + \nu$$

which is equal to

$$\begin{bmatrix} \mathbf{v} \\ \mathbf{a} \end{bmatrix}^{(k+1)} = \begin{bmatrix} \mathbf{I} & \Delta_t \mathbf{I} \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{a} \end{bmatrix}^{(k)} + \nu$$

and the measurement equation is

$$z(k) = \begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} x(k) + w$$

or

$$\frac{1}{\Delta_t} \begin{bmatrix} \mathbf{r} \\ \mathbf{t} \end{bmatrix}^{k} = \begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{a} \end{bmatrix}^{(k)} + w$$

as we directly measure $\mathbf{v}$ in Visual Odometry estimation procedure.

In these equations, $\mathbf{a}$ represents acceleration, $\mathbf{I}$ is the 6×6 identity matrix and ν and w represent Gaussian process and measurement noise, respectively.

We shall put the initial values as follows

$$\hat{x}(0|0) = \mathbf{0}_{6\times 1}$$

$$P(0|0) = \mathbf{I}_{12\times 1}$$

The reason behind choosing these values for them is that the filter will converge. The only important thing to consider is that if the value of $P(0|0)$ is significantly different, a larger value for process noise covariance is needed.

The next step is to see the results by noise variation. We shall start with the values in [10]

$$\nu \sim \begin{bmatrix} \mathcal{N}(0, \sigma_1 \mathbf{I}_{6\times6}) \\ \mathcal{N}(0, \mathbf{I}_{6\times6}) \end{bmatrix}$$

$$w \sim \mathcal{N}(0, \sigma_2 \mathbf{I}_{6\times6})$$

4.2 Extended Kalman Filter

This section deals with state estimation of nonlinear dynamic systems. Extended Kalman Filter (EKF) is a suboptimal estimator for this scenario [3]. We are interested in pose estimation of a device with a full possible range of motion. Estimating this pose with the Kalman filter would be impossible because the dynamics are non-linear.

The plant and measurement equations for these kind of non-linear systems can be written as

$$x(k+1) = f[k, x(k)] + \nu(k)$$

and

$$z(k) = h[k, x(k)] + w(k)$$

where $\nu(k)$ and $w(k)$ are the process and measurement noise respectively, which they are assumed to be zero-mean white Gaussian sequences.

An algorithm similar to the one from linear systems can be extended to the non-linear case .

Before proposing the algorithm we shall define the Jacobian of the vector f evaluated at the state $\hat{x}(k|k)$

$$f_x(k) \triangleq [\Delta_x f(k,x)']'|_{x=\hat{x}(k|k)} \triangleq \frac{\partial f}{\partial x}$$

Similarly,

$$f^i_{xx}(k) \triangleq [\Delta'_x \Delta_x f^i(k,x)]|_{x=\hat{x}(k|k)} \triangleq \frac{\partial^2 f^i}{\partial x^2}$$

is the Hessian of ith component of f.

One cycle of the EKF algorithm is as follows

State prediction, $\hat{x}(k+1|k) = f[k, \hat{x}(k|k)]$.

Evaluation of Jacobian, $F(k) = \frac{\partial f(k)}{\partial x}|_{x=\hat{x}(k|k)}$, $H(k+1) = \frac{\partial h(k+1)}{\partial x}|_{x=\hat{x}(k+1|k)}$.

State prediction covariance, $P(k+1|k) = F(k)P(k|k)F(k)' + Q(k)$.

Measurement prediction, $\hat{z}(k+1|k) = h[k+1, \hat{x}(k+1|k)]$.

Innovation covariance, $S(k+1) = H(k+1)P(k+1|k)H(k+1)' + R(k+1)$.

Filter gain, $W(k+1) \triangleq P(k+1|k)H(k+1)'S(k+1)^{-1}$.

The updated state estimate, $\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)\nu(k+1)$, where $\nu(k+1) \triangleq z(k+1) - \hat{z}(k+1|k) = \tilde{z}(k+1|k)$ is called the innovation or measurement residual.

Updated state covariance, $P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1)W(k+1)'$.

System details

As discussed before, a simple Kalman filter can not satisfy the scenario when the car turns. Therefor, we shall present the details of a new filter that deals with a nonlinear transition matrix and uses constant velocity and constant angular velocity

model. The state vector consists of current translation and orientation of the camera in unit quaternion representation [6]

$$x = \begin{bmatrix} \mathbf{t}^{WC} \\ \mathbf{q}^{WC} \\ \mathbf{V}^{W} \\ \mathbf{W}^{C} \end{bmatrix}$$

,

where $\mathbf{t}^{WC} = (t_x, t_y, t_z)^T$ denotes 3D translation of the camera in the world coordinate system, $\mathbf{q}^{WC} = (q_w, q_x, q_y, q_z)^T$ the unit quaternion showing the camera orientation relative to the world frame, $\mathbf{V}^{W}$ indicates the linear velocities of the camera along the coordinate axes of W and $\mathbf{W}^{C}$ is the angular velocities relative to the camera coordinate system C. Two coordinate systems that are used are the world coordinate system W and the camera coordinate system C.

We shall put the initial values as follows

$$\mathbf{t}^{WC} = (0, 0, 0)^T$$

$$\mathbf{q}^{WC} = (1, 0, 0, 0)^T$$

and the camera is considered to be unmoving $\mathbf{V}^{W} = \mathbf{W}^{C} = (0, 0, 0)^T$

$$P(0|0) = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & P_V \end{bmatrix}$$

where

$$P_V = \begin{bmatrix} \sigma^2_{V_0} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{V_0} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{V_0} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{W_0} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{W_0} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{W_0} \end{bmatrix}$$

The transition function f can be denoted as

$$x(k+1) = f[k, x(k)] = \begin{bmatrix} \mathbf{t}^{WC}(k+1) \\ \mathbf{q}^{WC}(k+1) \\ \mathbf{V}^{W}(k+1) \\ \mathbf{W}^{C}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{t}^{WC}(k) + \mathbf{V}^{W}(k)\Delta t \\ \mathbf{q}^{WC}(k) \times quat(\mathbf{W}^{C}(k)\Delta t) \\ \mathbf{V}^{W}(k) \\ \mathbf{W}^{C}(k) \end{bmatrix}$$

where $quat(\mathbf{W}^C(k)\Delta t)$ is the quaternion denotes the rotation of $\mathbf{W}^C(k)\Delta t$. As discussed in section 2.2.1, the equivalent angle-axis representation, $a =< \mathbf{a}, \alpha >$, of the Euler representation in 3D, $\mathbf{X} = (x, y, z)^T$, can be written as

$$a =< \mathbf{a}, \alpha >=< \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \alpha >=< \begin{bmatrix} \frac{x}{||\mathbf{X}||} \\ \frac{y}{||\mathbf{X}||} \\ \frac{z}{||\mathbf{X}||} \end{bmatrix}, ||\mathbf{X}|| >$$

Therefore, the quaternion q showing the same rotation can be written as

$$q = (\cos\frac{\alpha}{2}, \frac{a_x}{||\mathbf{a}||}\sin\frac{\alpha}{2}, \frac{a_y}{||\mathbf{a}||}\sin\frac{\alpha}{2}, \frac{a_z}{||\mathbf{a}||}\sin\frac{\alpha}{2})^T$$

The measurement equation shall be expressed as

$$z(k) = \begin{bmatrix} \mathbf{t}^{WC} \\ \mathbf{q}^{WC} \end{bmatrix} = H(k)x(k)$$

where

$$H = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}$$

Up to now, it is assumed that the translation and orientation of camera follows a constant velocity over time, but we need to denote a model for the noise representation. We shall assume that on average an undetermined acceleration is occurred with a Gaussian characteristics. This process noise model can be express as

$$\nu = \begin{bmatrix} \nu_V \\ \nu_W \end{bmatrix} = \begin{bmatrix} \nu_{a_V}\Delta t \\ \nu_{a_W}\Delta t \end{bmatrix}$$

where ν_{a_v} and ν_{a_W} denote the linear and angular acceleration process noise of zero mean and Gaussian distribution receptively which they are uncorrelated [6].

The process noise covariance $Q(k)$ can be written as

$$Q(k) = \frac{\partial f}{\partial \nu} P_\nu \frac{\partial f^T}{\partial \nu}$$

where

$$P_\nu = \begin{bmatrix} (\nu_{a_V}\Delta t)^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & (\nu_{a_V}\Delta t)^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & (\nu_{a_V}\Delta t)^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & (\nu_{a_W}\Delta t)^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (\nu_{a_W}\Delta t)^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (\nu_{a_W}\Delta t)^2 \end{bmatrix}$$

The smoothness of the camera motion is determined by the values of P_ν. The small parameters of P_ν is good for a very smooth motion with small accelerations, although, it is unable to follow camera's sudden rapid movements.

By implementing our EKF filter with a specific value of process noise, it is seen that one level of process noise does not satisfy our system's characteristics, as the straight and turning movement of the car needs different levels of noise.

Therefore, our system requires to adapt its process noise values depending on whether it is turning or not. To this end, in the next section an IMM filter which can adapt itself to certain types of uncertainties will be explained.

4.3 IMM

This section deals with the state estimation of systems with different models. These adaptive estimation algorithms can adapt themselves to certain types of uncertainties.

The multiple model (MM) algorithms assume that the system behaves according to one of a finite number of models (operating regimes). The models can be different in state dimensions, in noise levels or unknown inputs. One of the suboptimal approach, is the interacting multiple model (IMM) [3] which is presented here.

The steps required to calculate the IMM is as follows

Let $A(k1) = (M_1(k1), ..., M_i(k1), ...)$ be the mode set of the IMM estimator in the time interval $(k2, k1]$ and $A(k) = (M_1(k), ..., M_j(k), ...)$ be the mode set in the time interval $(k1, k]$.

Step 1 - Calculation of the mixing probabilities

The probability that mode M_i was in effect at $(k1)$ given that mode M_j is in effect at k conditioned on Z_1^{k-1} is

$$\begin{aligned}\mu_{i|j}(k-1|k-1) &= P\{M_i|M_j, Z_1^{k-1}\} \\ &= \frac{1}{C_j} P\{M_j|M_i, Z_1^{k-1}\} P\{M_i|Z_1^{k-1}\}\end{aligned}$$

The above mixing probabilities can be written as

$$\mu_{i|j}(k-1|k-1) = \frac{1}{C_j}[p_{ij}(S_{k-1})]\mu_i(k-1), \forall i, \forall j$$

where the normalizing constants are given by

$$C_j = \sum_{M_i \in A(k-1)} [p_{ij}(S_{k-1})]\mu_j(k-1), \forall j$$

Step 2 - IMM mixing

Mixed initial condition for the filter matched to $M_j(k\tau)$ is computed as

$$\hat{x}^{0j}(k-1|k-1) = \sum_{M_i \in A(k-1)} \hat{x}^i(k-1|k-1)\mu_{i|j}(k-1|k-1), \forall j$$

The corresponding covariance is given by

$$\begin{aligned} P^{0j}(k-1|k-1) = \sum_{M_i \in A(k-1)} & \mu_{i|j}(k-1|k-1)\{\hat{x}^i(k-1|k-1) \\ & + [\hat{x}^i(k-1|k-1) - \hat{x}^{0j}(k-1|k-1)] \\ & [\hat{x}^i(k-1|k-1) - \hat{x}^{0j}(k-1|k-1)]^T\}, \forall j \end{aligned}$$

Step 3 - Mode matched filtering

The estimate $\hat{x}^{0j}(k-1|k-1)$ and covariance $P^j(k-1|k-1)$ are used as input to the filter matched to M_j, which uses $z(k)$ to yield $\hat{x}^j(k|k)$ and $\hat{P}^j(k|k)$. The likelihood function corresponding to the above filtering processes are calculated by

$$\Lambda_j(k) = P[z(k)|M_j, Z_1^{k-1}], \forall j$$

Step 4 - Mode probability updating

The probability of model (M_j) being in effect during the time interval $(k, k1]$ given measurement data up to k can be calculated as

$$\begin{aligned}\mu_j(k|k) &= P\{M_j|Z_1^k\} \\ &= \frac{1}{C}P[z(k)|M_j, Z_1^{k-1}]P[M_j|Z_1^{k-1}] \\ &= \frac{1}{C}\Lambda_j(k)\mu_j(k|k-1)\end{aligned}$$

where $\mu_j(k|k-1)$ is

$$\mu_j(k|k-1) = \sum_{M_i \in A(k-1)} [p_{ij}(S_{k-1})]\mu_i(k-1)$$

Step 5 - State estimate and covariance combination

The final state estimate and covariance can be written as

$$\hat{x}(k|k) = \sum_{M_j \in A(k)} \hat{x}^j(k|k)\mu_j(k|k)$$

$$P(k|k) = \sum_{M_j \in A(k)} \{P^j(k|k) + [\hat{x}^j(k|k) - \hat{x}(k|k)][\hat{x}^j(k|k) - \hat{x}(k|k)]^T\}$$

More detailed on how these steps are driven can be found in [3].

System details

Our system consists of 2 modes which uses the same system transfer function as the one used in EKF approach but with different process noise. Designed is based on the fact that the system characteristic is different when the vehicle is going straight from when the vehicle is making a turn.

The first mode, consists of a the mode when the vehicle is moving in a straight line and no turning occurs. In this scenario we need a very smooth motion of camera, hence smaller values for the linear and angular acceleration noises are needed. In the implementation, this value is in the order of 10^{-4} for ν_{a_V} and 10^{-6} for ν_{a_W}.

The second mode, consists of a the mode when the vehicle is turning. In this scenario we need to cope with rapid acceleration and to follow camera's sudden movements. Therefore, bigger values for the linear and angular acceleration noises are needed. In the implementation, this value is in the order of 10^{-2} for ν_{a_V} and 10^{-3} for ν_{a_W}.

Chapter 5

Simulation Studies and Results

In this chapter, the results of experimental evaluation of our system are presented and discussed.

Dataset

We tested the proposed approach on the well-known KITTI dataset [9]. These datasets contain high-quality sequences collected from a moving vehicle. Also, these dattasets provided the ground truth trajectory, which is recorded by the output of the GPS/IMU sensors[8].

We consider a calibrated setup and rectified input images, as this is regarded as the standard case and make the computations easier. We further know the camera's height from the ground and how much it faces downwards (pitch).

First of all, we examine 5 different sequences from KITTI 1-10, as their ground truth information is available. The sequences that have been used are shown in figure 5.1. We chose different scenario from the simplest case where the car goes straight ,figure 5.1a, to the case where the car turns frequently ,figure 5.1e.

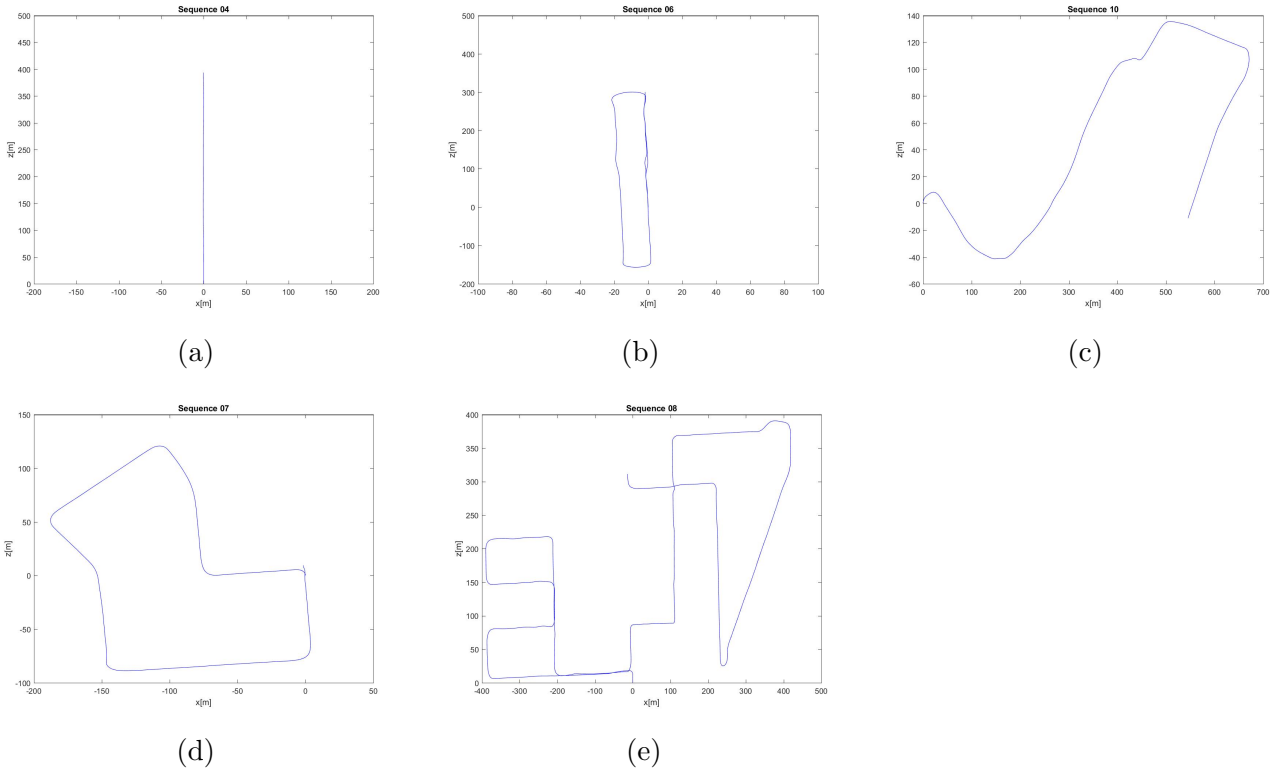

Figure 5.1: Ground truth trajectory of some of the KITTI dataset

Error Metric

One way of evaluating VO is to calculate the error of the trajectory end-point. But this method can be ambiguous, as it depends on the point in time where the error has occurred (errors earlier in the sequence lead to larger end-point errors) [9]. Here we calculate the average of all relative relations at a fixed distance for translation and rotation separately. This error metric can be formulated as follows

$$E_{rot}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j \in \mathcal{F})} [(\hat{P}_j ic \hat{P}_i) ic (P_j ic P_i)]$$

$$E_{trans}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j \in \mathcal{F})} ||(\hat{P}_j ic \hat{P}_i) ic (P_j ic P_i)||_2$$

where $\mathcal{F}$ is a set of frames of a specific sequence, $\hat{P}$ and P are the estimated and true camera pose respectively, and ic indicates inverse compositional operation [20].

Errors are measured in percent (for translation) and in degree per meter (for rotation).

Results

In this section, we compared our proposed filtering with the already existing algorithm VISOM; the monocular version of the work in [10], and consider it as our counterpoint.

For each sequence of the 5 different scenario from sequence 1-10, we shall put the results of estimated translation and rotation parameters as well as the estimated trajectory with comparison by the available ground truth information.

The results for sequence 04 are presented in figures 5.2, 5.3 and 5.4. Looking at figure 5.4 we can see the estimated trajectory after filtering is more similar to the ground truth.

The results for the rest of sequences 06, 10, 07 and 08 are presented in figures 5.5, 5.6 and 5.7 to figures 5.17, 5.18 and 5.19. They all shows improvement in terms of accuracy.

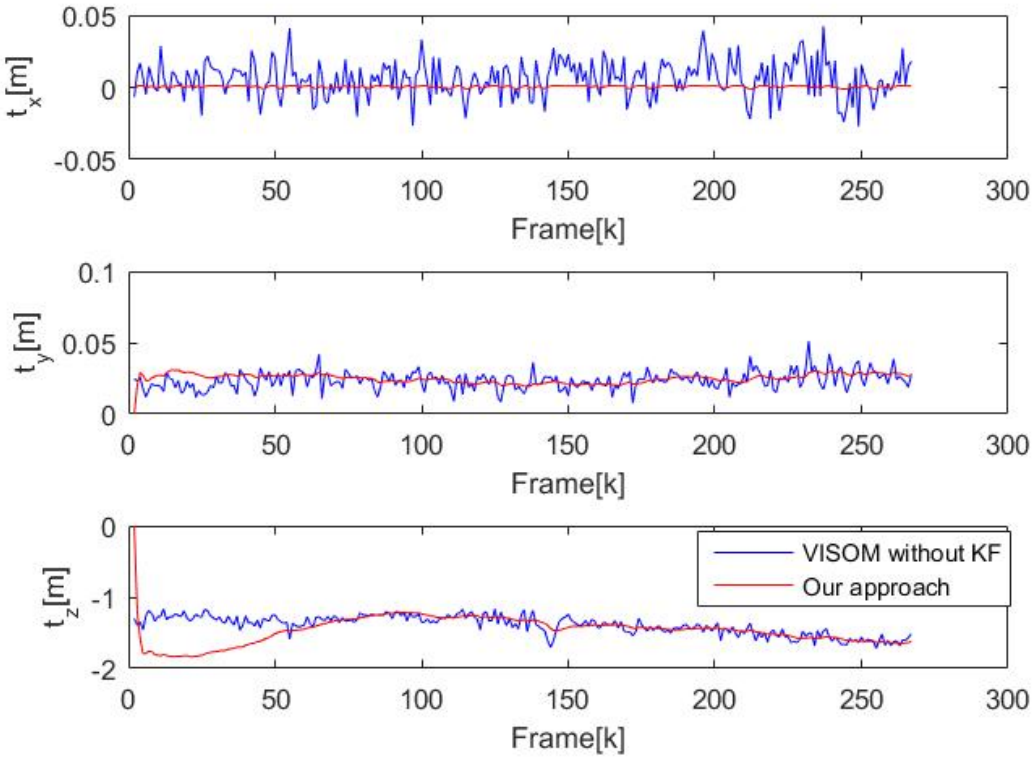

Figure 5.2: Translation for sequence 04

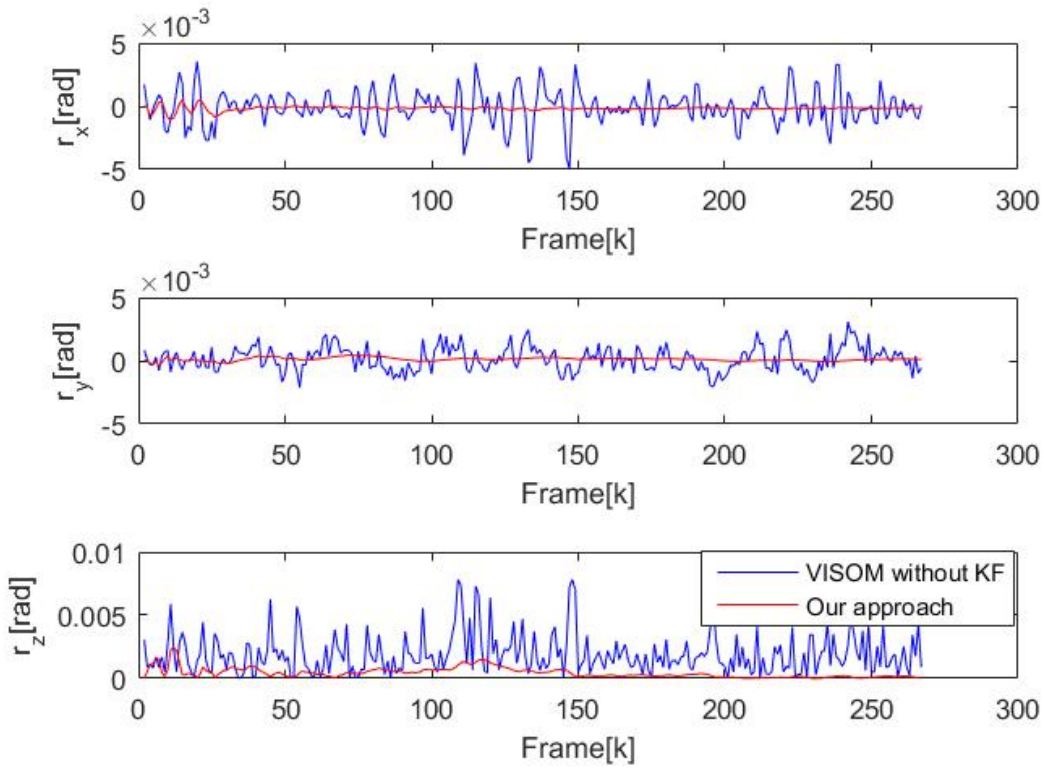

Figure 5.3: Rotation for sequence 04

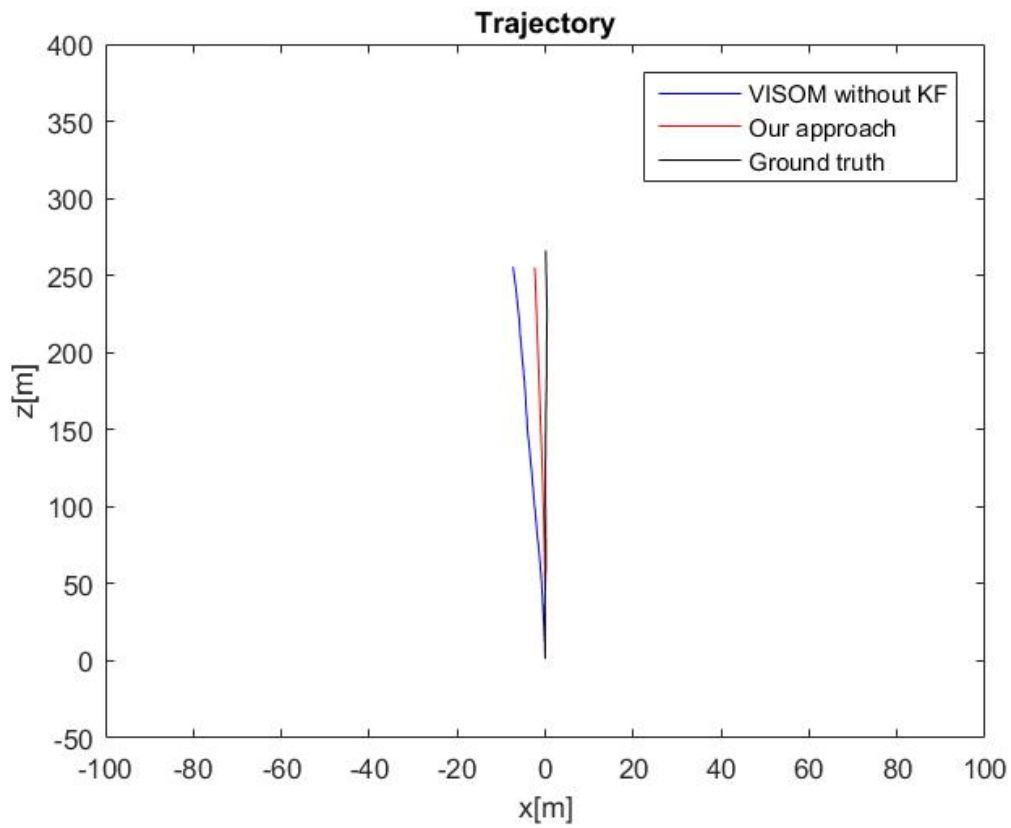

Figure 5.4: Trajectory for sequence 04

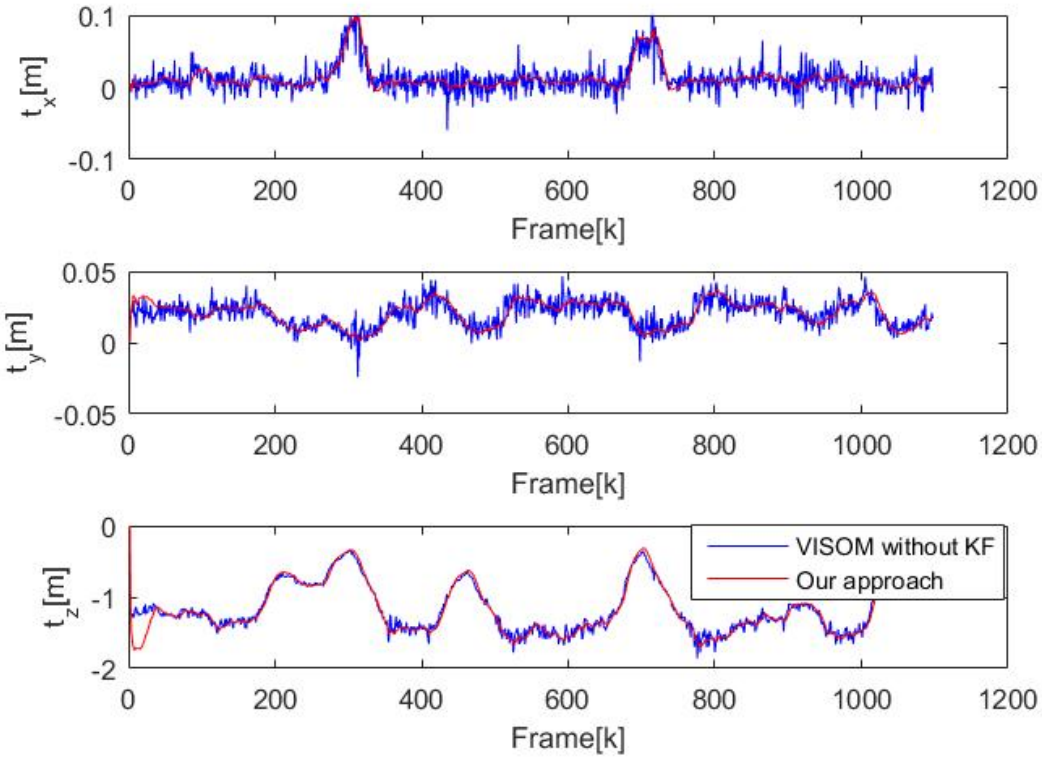

Figure 5.5: Translation for sequence 06

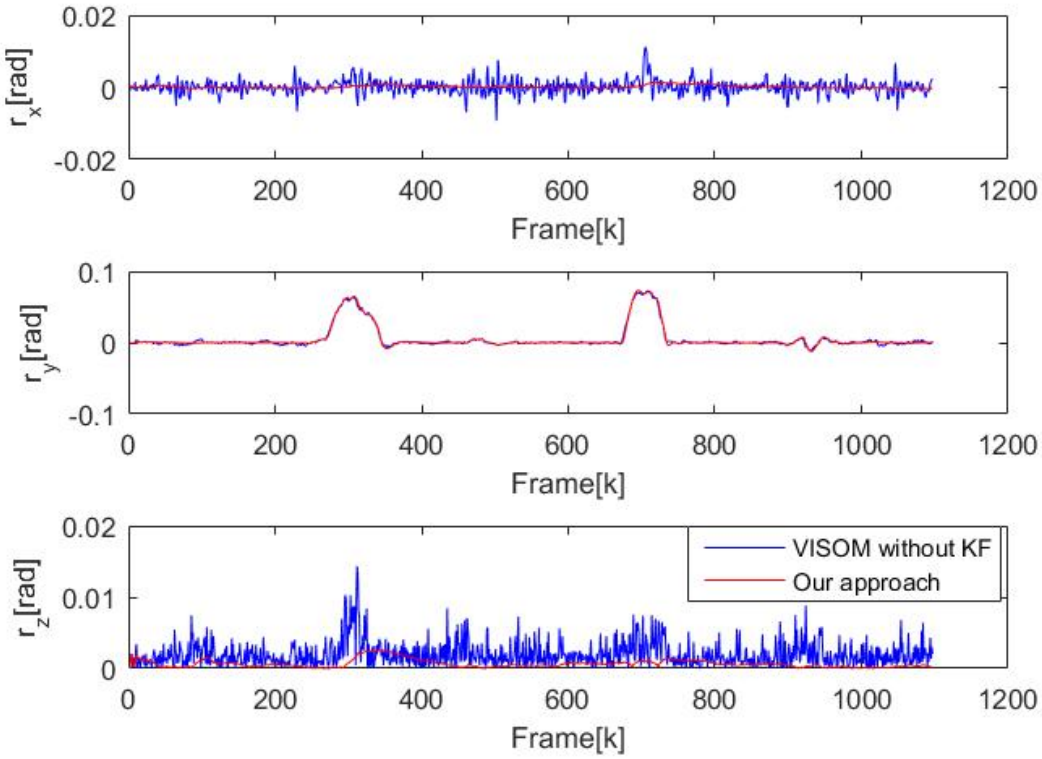

Figure 5.6: Rotation for sequence 06

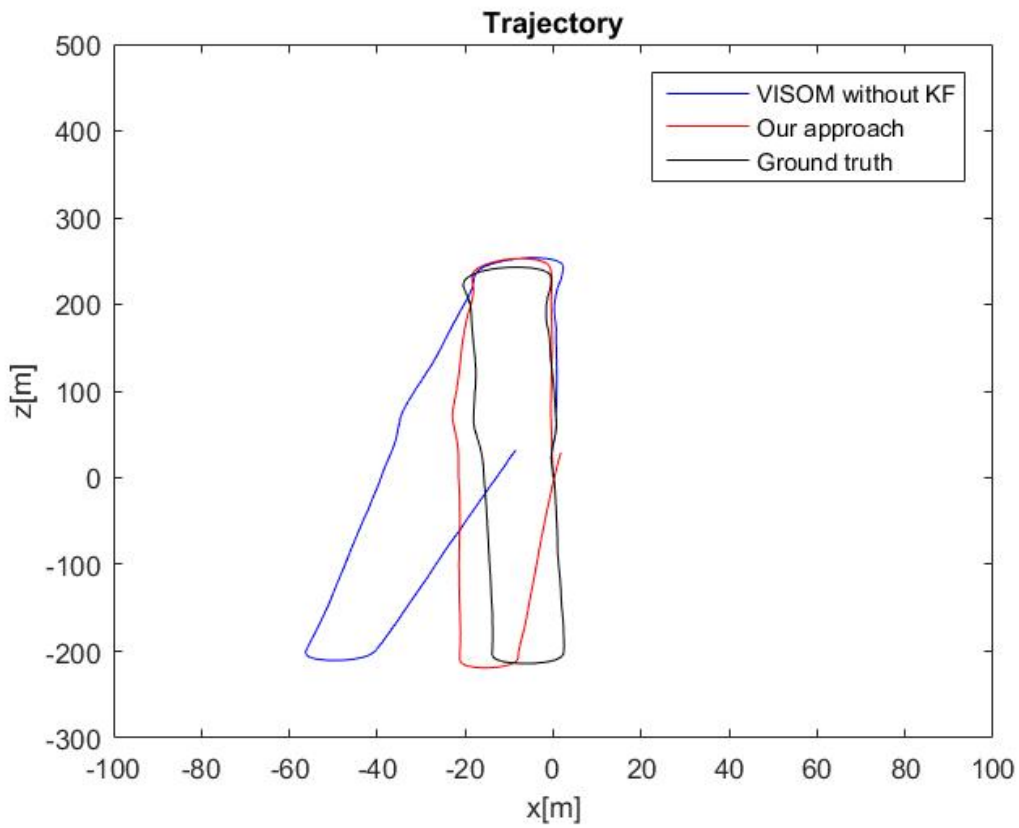

Figure 5.7: Trajectory for sequence 06

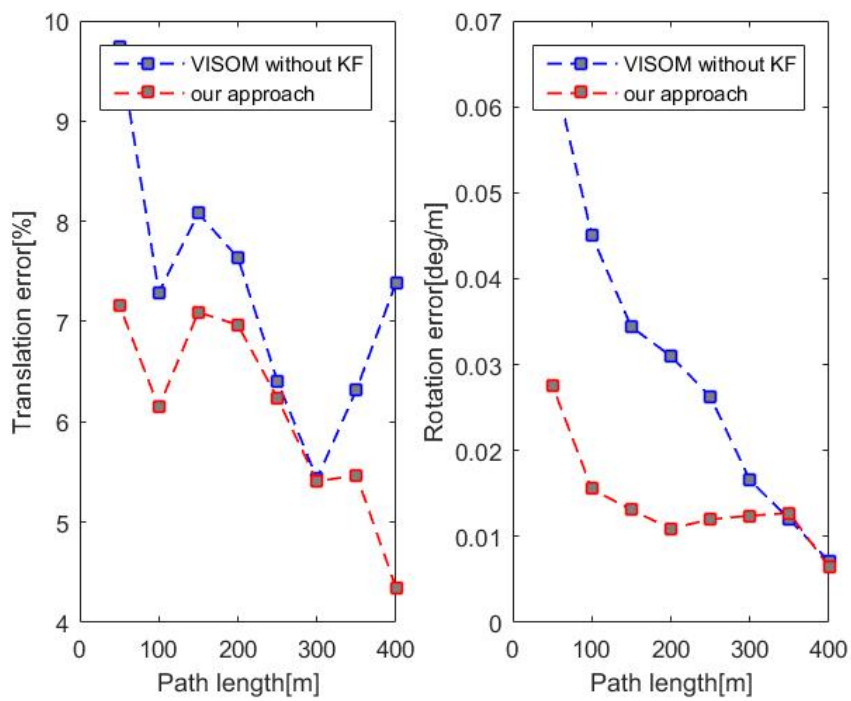

Figure 5.8: Translation and Rotation error 06

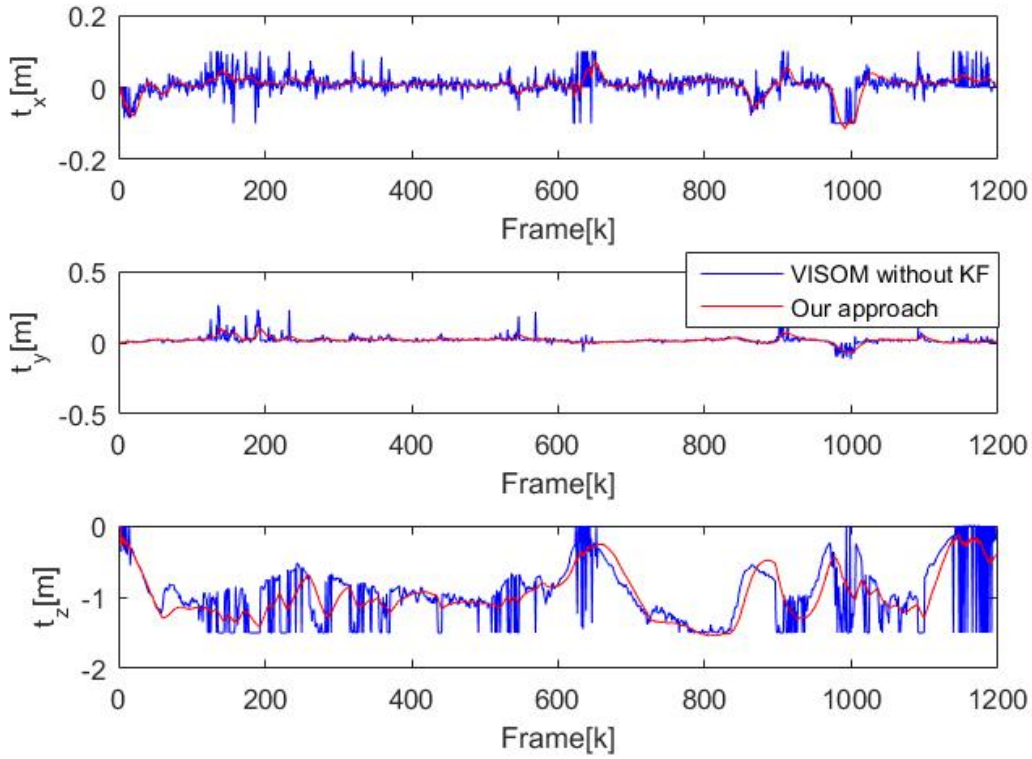

Figure 5.9: Translation for sequence 10

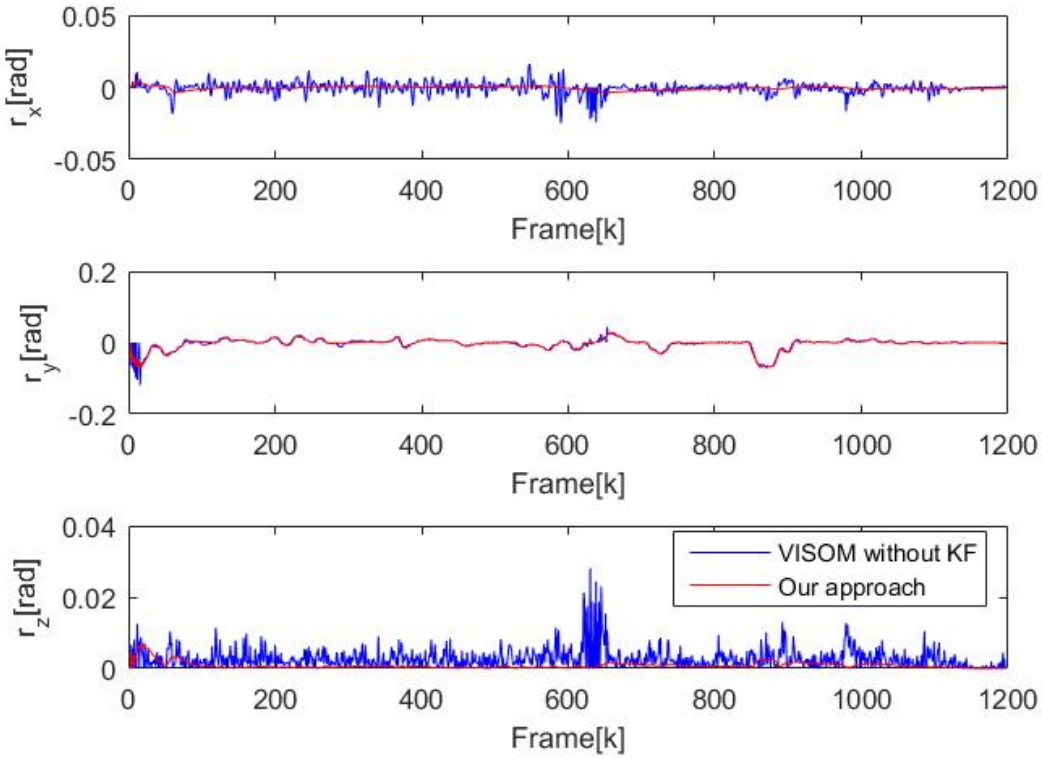

Figure 5.10: Rotation for sequence 10

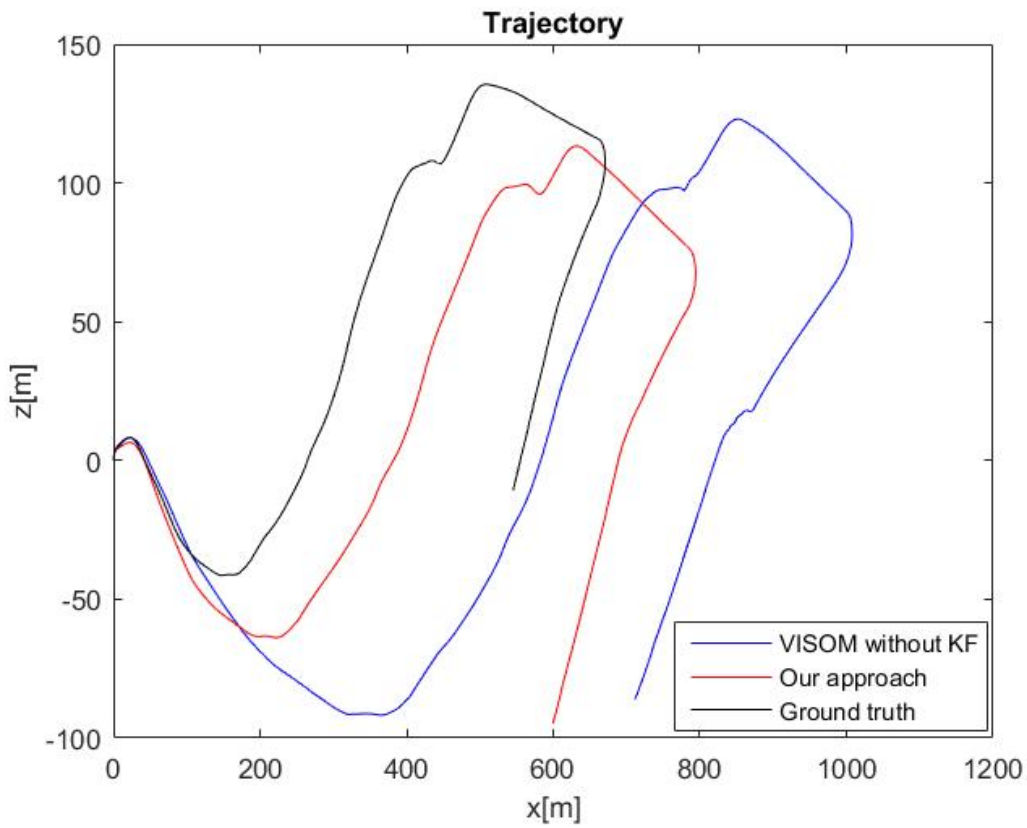

Figure 5.11: Trajectory for sequence 10

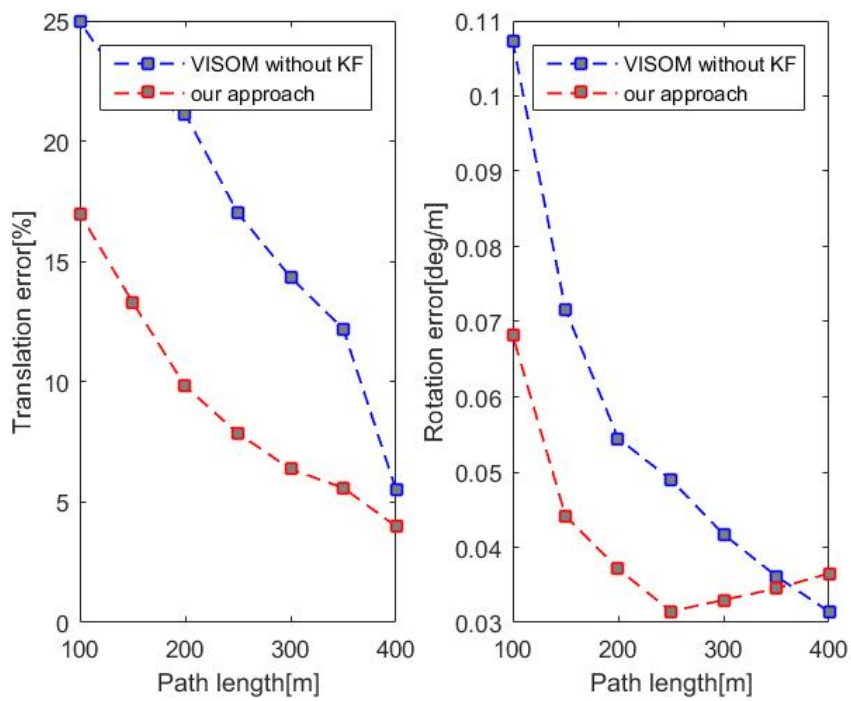

Figure 5.12: Translation and Rotation error 10

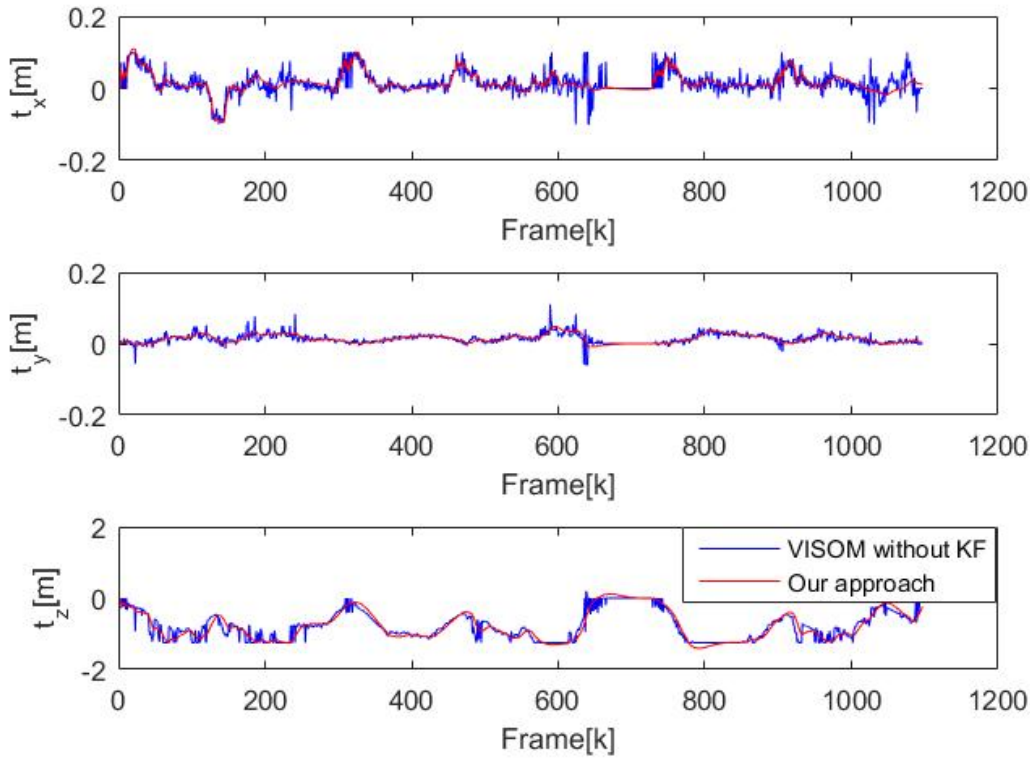

Figure 5.13: Translation for sequence 07

By applying the discussed error metric, we obtain the results in table 5.1. It can be seen that the average translational and rotation error has declined.

More comparison

Furthermore, we compare the proposed method with the algorithms in [10] but with another scale estimation method, which is the work in [39] using a novel cue combination framework for ground plane estimation, which is more computationally complex than the implemented scale estimation method of our approach.

The following results in figures 5.22, 5.23 and 5.23 are extracted by using sequences

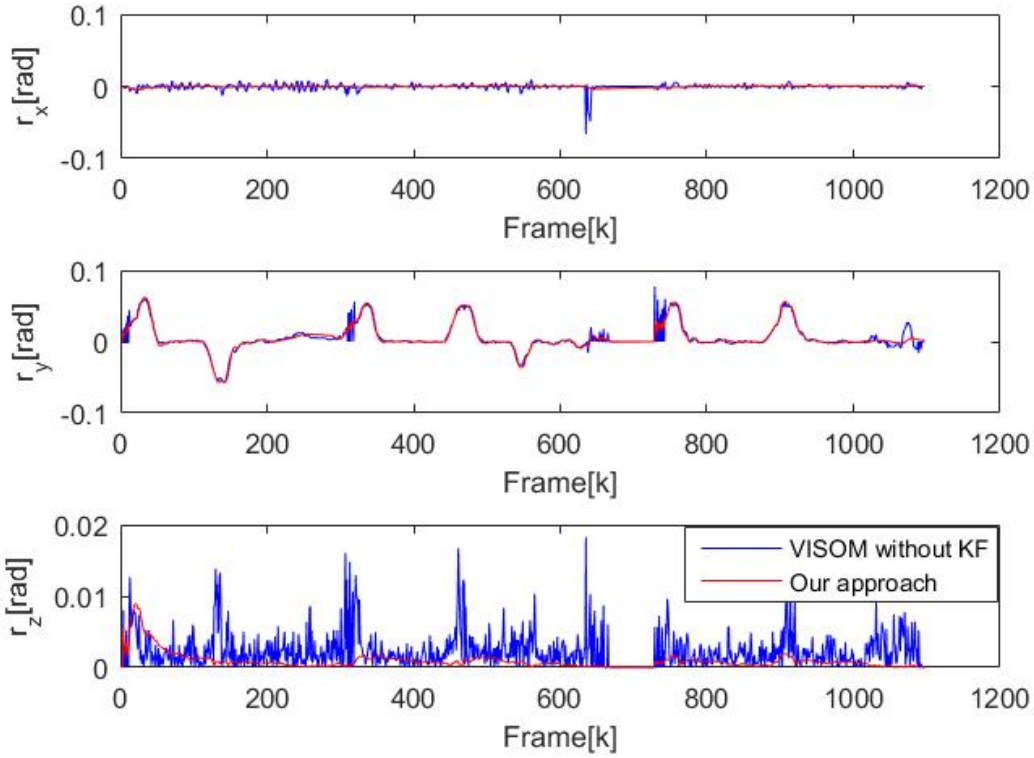

Figure 5.14: Rotation for sequence 07

11-15 of the KITTI data set.

As shown in table 5.1, there is improvement in terms of accuracy by applying our approach, but the errors can be further decreased by utilizing another scale estimation method such as the one in [39]. We can conclude that our method is compatible with the method using another scale estimation approach, which is computationally more complex than the scale estimation approach in VISOM [10].

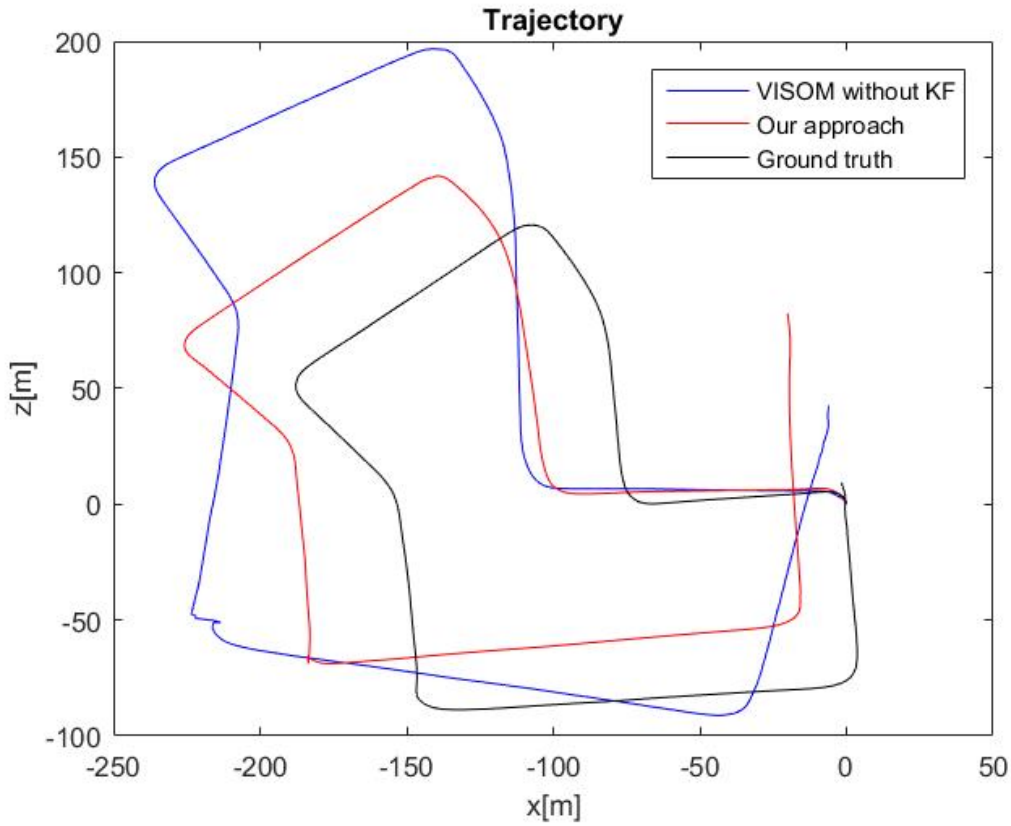

Figure 5.15: Trajectory for sequence 07

	VISOM without KF		Our Approach	
Sequence	Trans. err.%	Rot. err. [deg/m]	Trans. err.%	Rot. err. [deg/m]
04	2.86	.0051	2.76	.0077
06	7.38	.029	6.12	.0166
10	16.71	.0559	9.14	.0407
07	19.41	.055	15.46	.0476
08	20.66	.39	12.28	.076
Average	13.39	.107	9.15	.0377

Table 5.1: Average error for different sequences

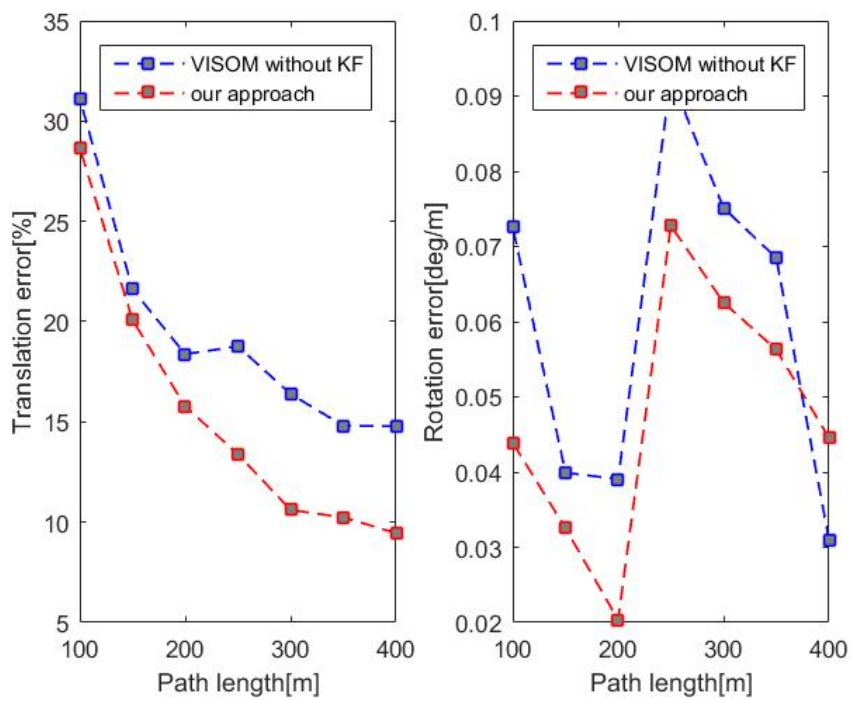

Figure 5.16: Translation and Rotation error 07

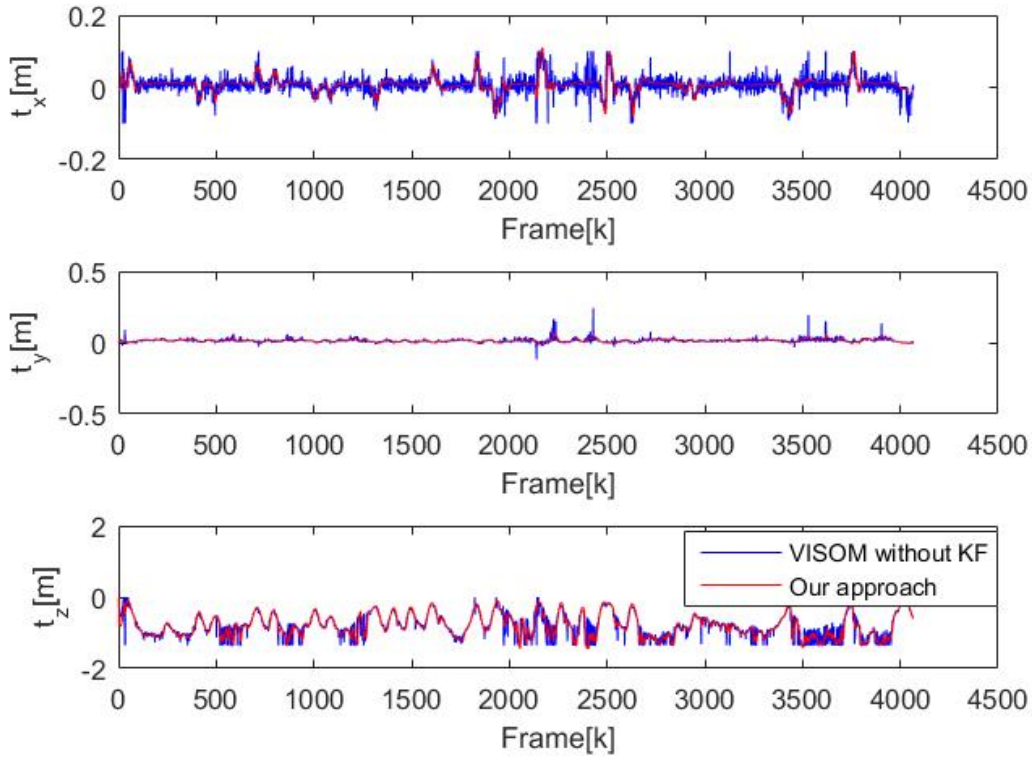

Figure 5.17: Translation for sequence 08

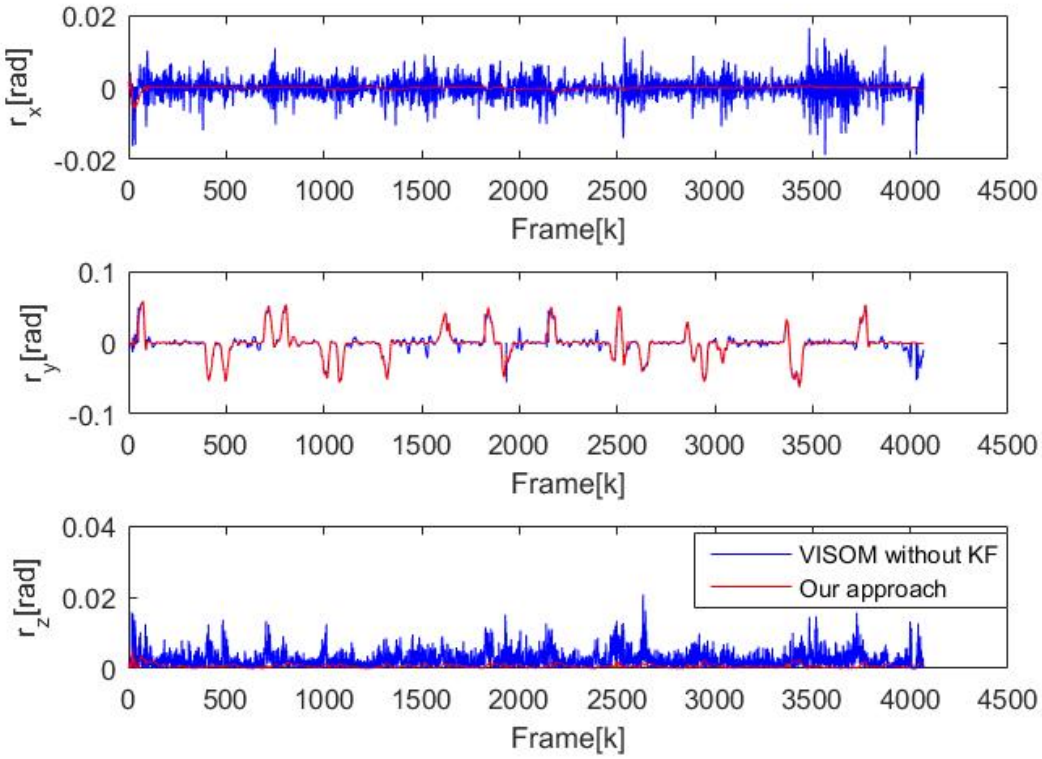

Figure 5.18: Rotation for sequence 08

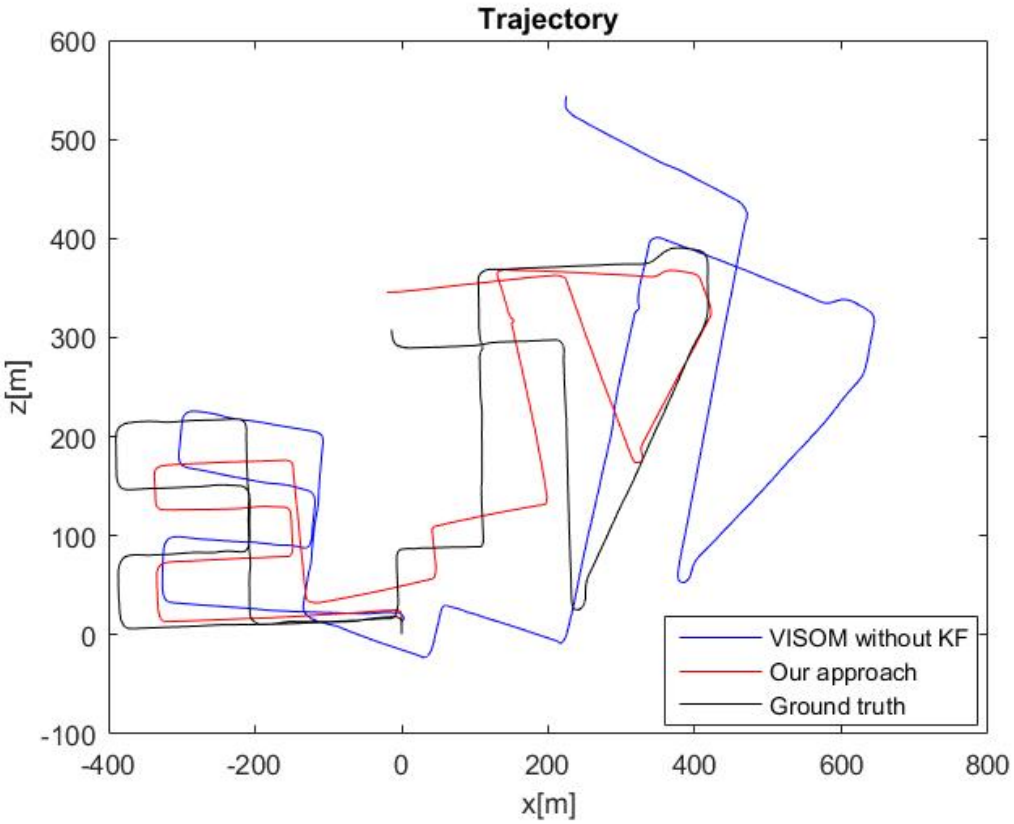

Figure 5.19: Trajectory for sequence 08

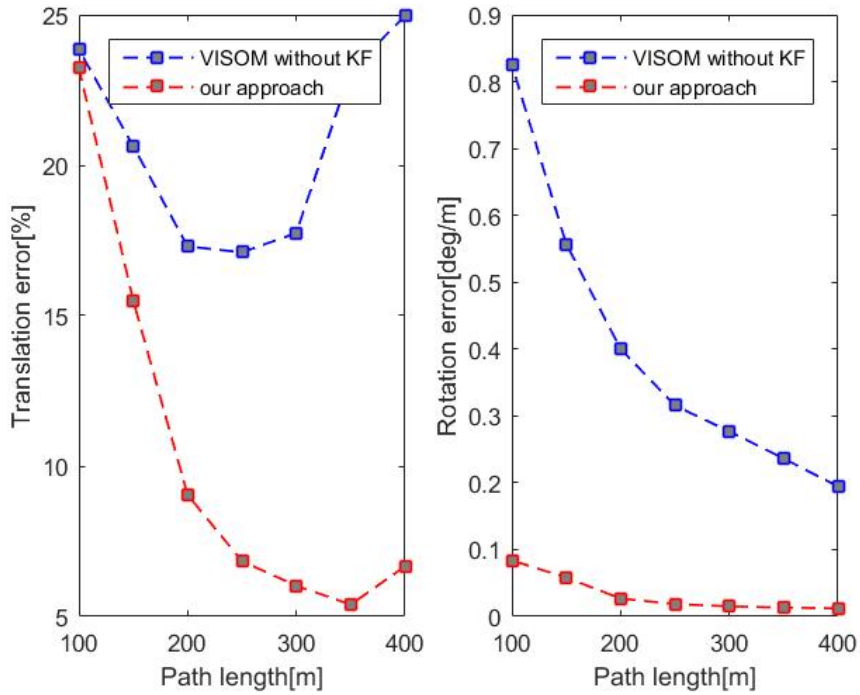

Figure 5.20: Translation and Rotation error 08

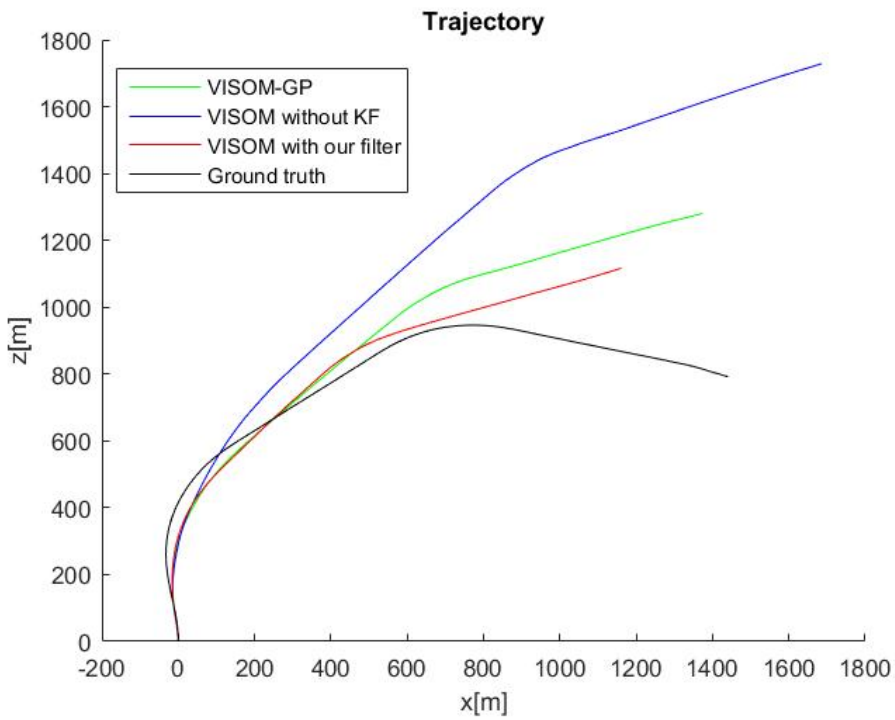

Figure 5.21: Trajectory for sequence 12

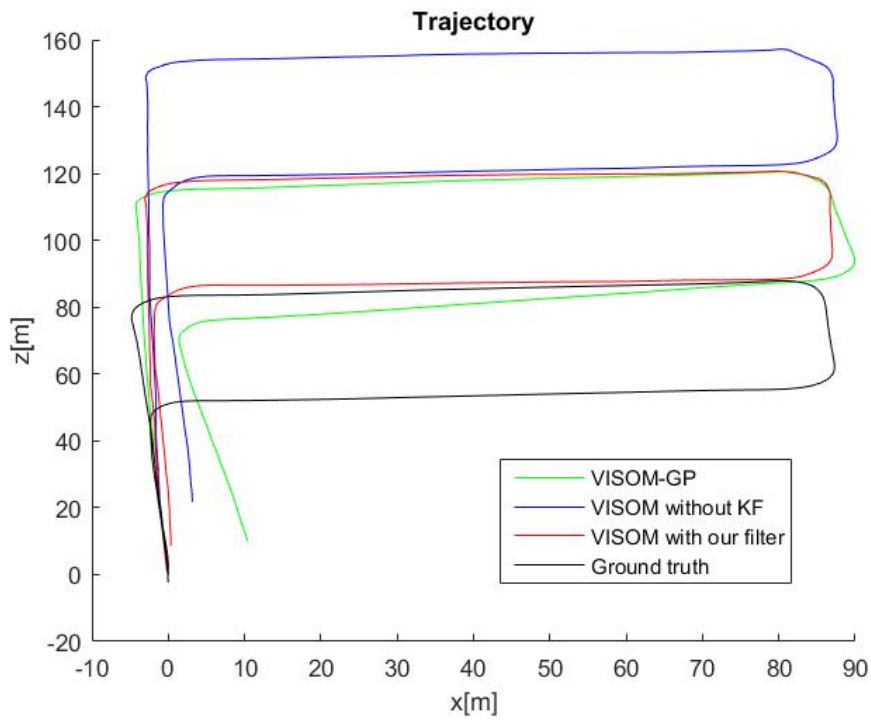

Figure 5.22: Trajectory for sequence 14

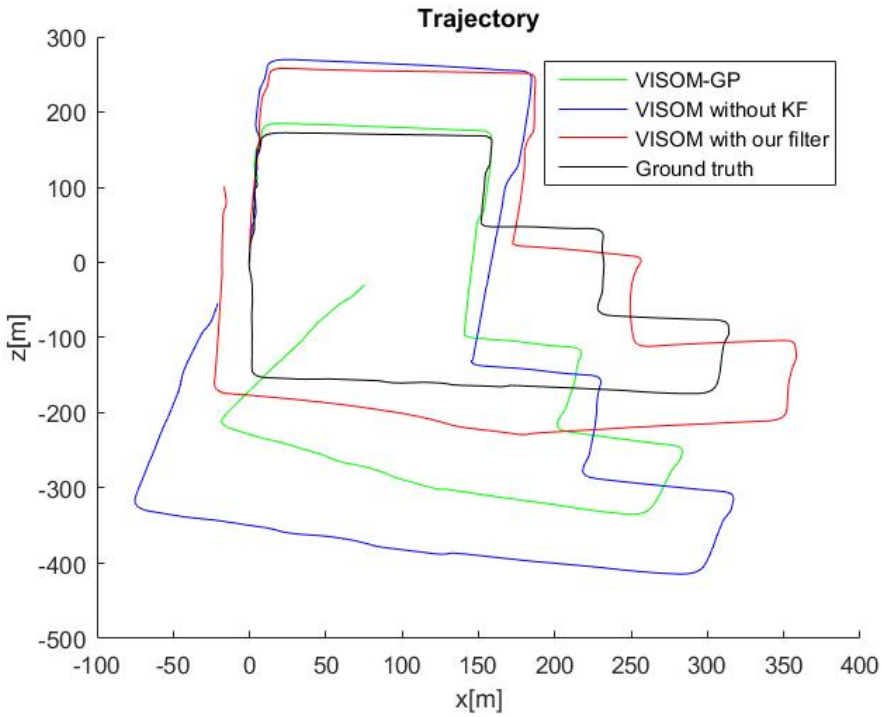

Figure 5.23: Trajectory for sequence 15

Chapter 6

Conclusions and Future Works

In this thesis, a monocular Visual Odometry system has been developed and tested on different datasets and the outputs have been compared with the available ground truth information to analyze the precision of the system. This system is capable of estimating the 3D position of a ground vehicle robustly and in real time.

As demonstrated in chapter 3, one of the main challenges of monocular VO is the ambiguity of scale estimation as there is no fixed base-line to extract 3D information. To tackle this problem, it was assumed that the ground is locally planar and the height of the mounted camera on the car from the ground is fixed and known.

Furthermore as discussed in chapter 4, in order to improve the VO estimation and to help other stages of VO process an effective filtering approach is utilized. By applying an efficient filter we can have better results with the same amount of computational complexity. This filtering also can be applied in any other aspects when we are dealing with a moving camera and we are interested in the ego motion and the own displacement of the camera. Moreover, it is shown that an IMM filtering can address the needs of this specific application, as the movement of a ground vehicle

is different depending on different scenarios.

Chapter 5 illustrates the results of simulation on the KITTI dataset and it is shown that our system's accuracy improved compared to what is considered to be one of the best state-of-the-art monocular Visual Odometry system.

Future Work

In order to improve the results of this Visual Odometry system, there are several interesting ideas that could be investigated.

One of the idea is to utilize a different scale estimation method in our VO system to see whether it can compete with the current best performing stereo VO algorithms.

We further can extend our filtering approach to the case of binocular cameras and use the outputs of the filtered states to improve other stage of VO e.g. feature matching and tracking. This would help to reduce the computational complexity by decreasing refinement methods such as bundle adjustment and RANSAC.

Bibliography

[1] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5(1):1897, 2016.

[2] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987.

[3] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[4] Javier Civera, Oscar G Grasa, Andrew J Davison, and JMM Montiel. 1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631, 2010.

[5] Laura A Clemente, Andrew J Davison, Ian D Reid, José Neira, and Juan D Tardós. Mapping large loops with a single hand-held camera. In *Robotics: Science and Systems*, volume 2, 2007.

[6] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse.

Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[7] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

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

[10] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968. Ieee, 2011.

[11] Roland Goecke, Akshay Asthana, Niklas Pettersson, and Lars Petersson. Visual vehicle egomotion estimation using the fourier-mellin transform. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 450–455. IEEE, 2007.

[12] José Jesús Guerrero, Ruben Martinez-Cantin, and Carlos Sagüés. Visual mapless navigation based on homographies. *Journal of Field Robotics*, 22(10):569–581, 2005.

[13] Christopher G Harris and JM Pike. 3d positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988.

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

[15] Qifa Ke and Takeo Kanade. Transforming camera geometry to a virtual downward-looking camera: Robust ego-motion estimation and ground-layer detection. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2003.

[16] Bernd Manfred Kitt, Jörn Rehder, Andrew D Chambers, Miriam Schonbein, Henning Lategahn, and Sanjiv Singh. Monocular visual odometry using a planar road model to solve scale ambiguity. 2011.

[17] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[18] Georg Klein and David Murray. Improving the agility of keyframe-based slam. *Computer Vision–ECCV 2008*, pages 802–815, 2008.

[19] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2969–2976. IEEE, 2011.

[20] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[21] Simon Lacroix, Anthony Mallet, Raja Chatila, and Laurent Gallo. Rover self localization in planetary-like environments. In *Artificial Intelligence, Robotics and Automation in Space*, volume 440, page 433, 1999.

[22] Bojian Liang and Nick Pears. Visual navigation using planar homographies. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 205–210. IEEE, 2002.

[23] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[24] Larry Matthies. *Dynamic stereo vision*. PhD thesis, Carnegie Mellon University, 1989.

[25] Larry Matthies and STEVENA Shafer. Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248, 1987.

[26] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. Rslam: A system for large-scale mapping in constant-time using stereo. *International journal of computer vision*, 94(2):198–214, 2011.

[27] Michael J Milford and Gordon F Wyeth. Single camera vision-only slam on a suburban road network. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3684–3689. IEEE, 2008.

[28] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1980.

[29] Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Accurate non-iterative o (n) solution to the pnp problem. In *Computer vision, 2007. ICCV 2007. IEEE 11th international conference on*, pages 1–8. IEEE, 2007.

[30] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 850–855. IEEE, 2006.

[31] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[32] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. Ieee, 2004.

[33] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[34] Davide Scaramuzza. 1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints. *International journal of computer vision*, 95(1):74, 2011.

[35] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[36] Davide Scaramuzza, Friedrich Fraundorfer, Marc Pollefeys, and Roland Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1413–1419. IEEE, 2009.

[37] Davide Scaramuzza, Friedrich Fraundorfer, and Roland Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4293–4299. Ieee, 2009.

[38] Davide Scaramuzza and Roland Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE transactions on robotics*, 24(5):1015–1026, 2008.

[39] Shiyu Song and Manmohan Chandraker. Robust scale estimation in real-time monocular sfm for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1566–1573, 2014.

[40] Shiyu Song, Manmohan Chandraker, and Clark C Guest. Parallel, real-time monocular visual odometry. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4698–4705. IEEE, 2013.

[41] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2, 2010.

[42] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[43] Jean-Philippe Tardif, Yanis Pavlidis, and Kostas Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2531–2538. IEEE, 2008.

[44] Hui Wang, Kui Yuan, Wei Zou, and Qingrui Zhou. Visual odometry based on locally planar ground assumption. In *Information Acquisition, 2005 IEEE International Conference on*, pages 6–pp. IEEE, 2005.

[45] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009.