Estimating and Correcting Bias in Stereo Visual Odometry

by

Sara Farboud-Sheshdeh

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

Estimating and Correcting Bias in Stereo Visual Odometry

Sara Farboud-Sheshdeh

Master of Applied Science

Graduate Department of Electrical and Computer Engineering

University of Toronto

2014

Stereo visual odometry (VO) is a common technique for estimating a camera's motion; features are tracked across frames and the pose change is subsequently inferred. This position estimation method can play a particularly important role in environments in which the global positioning system (GPS) is not available (e.g., Mars rovers). Recently, some authors have noticed a bias in VO position estimates that grows with distance travelled; this can cause the resulting position estimate to become highly inaccurate. In this thesis, two effects are identified at play in stereo VO bias: first, the inherent bias in the maximum-likelihood estimation framework, and second, the disparity threshold used to discard far-away and erroneous stereo observations. In order to estimate the bias, the sigma-point method (with modification) combined with the concept of bootstrap bias estimation is proposed. Based on simulations, the novel sigma-point method achieves similar accuracy to Monte Carlo experiments, but at a fraction of the computational cost. The approach is validated through simulations.

Dedication

To my best friend, my love, my husband Majid.

Acknowledgements

I would like to thank my supervisors Prof. Raymond Kwong and Prof. Tim Barfoot for giving me the confidence to go forward and complete my graduate studies. I appreciate their patience and endless enthusiasm about the subject that turned all the improvements into my first academic research achievement. I am also very thankful for their continuous support and valuable comments on my thesis.

I would also like to express my gratitude to the chair of my thesis defense committee, Prof. Ben Liang, and my examiners Prof. Walter Wonham and Prof. Lacra Pavel for their helpful suggestions on my thesis.

In addition, I want to thank all the professors and my friends in the Systems Controls Group (SCG) and my other friends at the Autonomous Space Robotics Lab (ASRL) for their encouragement and the time they spent on discussing academic problems with me.

Of course, this journey would not have been possible without the effort and encouragement of my parents, Farah and Reza, and my only sister, Nazanin.

Lastly, I thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting me with the CGSM award in the Robotics discipline.

Contents

List of Figures

Notation

a:	Symbols in this font are real scalars.
$\mathbf{a}$:	Symbols in this font are real column vectors.
$\mathbf{A}$:	Symbols in this font are real matrices.
$\sim \mathcal{N}(\mathbf{a}, \mathbf{B})$:	Normally distributed with mean $\mathbf{a}$ and covariance $\mathbf{B}$.
$E[\cdot]$:	The expectation operator.
$(\cdot)_k$:	The value of a quantity at time step k.
$(\cdot)_j$:	The value of a quantity associated with landmark j.
$(\cdot)_i$:	The value of a quantity at iteration i.
$(\cdot)_\mathrm{t}$:	The true value of a quantity.
$\hat{(\cdot)}$:	The estimate of a quantity.
$\bar{(\cdot)}$:	The nominal value of a quantity.
$(\cdot)^\times$:	The cross-product operator which produces a 3×3 skew-symmetrix matrix from a 3×1 vector.
$(\cdot)^\boxplus$:	A *SE(3)* operator which produces a 4×4 matrix from a 6×1 vector.
$(\cdot)^\boxminus$:	A homogeneous point operator converting a 4×1 vector to a 4×6 matrix.
$\mathbf{1}$:	The identity matrix.
$\mathbf{0}$:	The zero matrix.
$\mathrm{tr}(\cdot)$:	The trace of a matrix.

Chapter 1

Introduction

Regardless of how basic or advanced a mobile robot is, whether its goal is to track a pre-defined trajectory or to traverse a terrain to reach a certain destination, it should be equipped with appropriate technologies in order to be able to navigate in its environment. Mobile robot navigation has been a popular research topic with applications in different areas from underground to space. The problem of navigation has been studied in the past using simple formulas and recently by proposing more complicated techniques. *Dead reckoning* in navigation is the process of estimating the vehicle's current position using the past known position and information about a vehicle's course and speed over a period of time. *Wheel odometry* is a simple technique of dead reckoning that uses encoder readings to calculate the number of wheel turns. Using knowledge about wheel diameters, this enables the distance travelled to be estimated. In wheel odometry, the noise sources are basically errors in the estimate of wheel diameters and slippage caused by fast turning or slippery ground. Wheel odometry measurements were used in the recent Mars Exploration Rover (MER) mission; however, they were unreliable due to the errors that wheel slip had caused [67]. Overall, for long distances the error terms add up incrementally which leads to a

poor estimate of the vehicle's position. Vision-based positioning is a more complicated technique for mobile robot navigation that relies on optical sensors. *Visual odometry* (VO) is the process of estimating the position and orientation of a vehicle using images captured by a camera attached to it. VO is a more advanced odometry technique which does not require dealing with slippage error, thus providing more accurate estimates. It is worthwhile to mention that the term VO has been introduced by Nister [79] in 2004; however, the origin of the visual odometry concept goes back to Moravec [77] (1980) and was later improved by Matthies and Shafer [70] in 1986.

In a visual odometry system, the vehicle's motion is estimated using two successive pairs of stereo images with no prior knowledge of the scene. Since the Mars Rover landing in 2004, the NASA Mars exploration program team have decided to use visual odometry to update the position estimate obtained by wheel odometry whenever they encountered slippery environments. The results showed improved position estimation of the Mars Exploration Rover vehicles [18]. Visual odometry plays a particularly important role in environments in which the global positioning system (GPS) is not available. Therefore, VO is a key technique for the navigation system of underwater vehicles [13], space rovers [48, 67], and underground robots in mining applications [39].

Although stereo visual odometry is a well-established technique, research is ongoing to improve its performance. Recently, some authors investigating VO have noticed a bias in the estimate that grows as the vehicle travels, resulting in an erroneous estimate. Dubbelman and Groen [29] point out the existence of bias in stereo-based motion estimation. They identify the distribution of landmarks and incorrect modelling of uncertainties in landmark positions as two sources of bias; they are able to partially compensate for bias through calibration methods.

Tardif et al. [92] mention the bias in stereo visual odometry systems and attempt

to produce a robust navigation system by reducing the bias using an inertial navigation filter; they do not estimate the bias itself. Similarly, Lambert et al. [61] limit the error growth in the VO estimate by integrating a sun sensor and an inclinometer for attitude correction, but do not actually estimate bias. However, Lambert [62] does mention that errors in visual odometry are biased and suggests that the disparity threshold plays a role in the bias. Dubbelman et al. [30] propose a method to model and compensate for bias using projective geometry; they mention that the camera model used in their research is an approximation and identify this approximation as the source of bias. They use trajectory calibration to estimate parameters of a bias model and use it to compensate for bias. Rehder et al. [85] also propose a solution for bias estimation based on brute-force Monte Carlo simulation, which could be computationally expensive for large numbers of samples. They study this problem in a one-dimensional space by estimating the bias existing in the norm of a vector that describes the translation of the camera in one time step. Their experimental results demonstrate that the actual motion is underestimated by VO, but is corrected by their bias estimation technique. Our research is related to these works, but we have found that first, even if the camera model is correct, there is an inherent bias in the maximum-likelihood framework used to estimate motion in VO, and, second, that the disparity threshold commonly used to throw away poor stereo observations also plays an important role in determining the bias.

Bias estimation is well studied in the statistics literature. Bias has different types such as *selection bias* [20] and the *bias of an estimator*. The bias of an estimator is defined as the difference between the expected value of the estimator and the true value of the unknown parameter to be estimated by the estimator [25]. In statistical sciences, the *bootstrap* technique is used to quantify uncertainty of the estimates including the bias of an estimator [50, 19]. The bootstrap method was originally

proposed by Efron [31] and developed by Efron and Gong [34], Efron and Tibshirani [35], Diaconis and Efron [28], and in the monograph [32]. The bootstrap methodology is not used in its original form in this thesis; however, the idea behind it will be borrowed to create a method to estimate bias in VO problems.

The main objective of this thesis is to propose a novel approach that can estimate and correct bias in VO in a computationally inexpensive manner without losing accuracy. In particular, we will not employ any other sensors except for a stereo camera in our approach and thus the only input to our algorithm would be a set of stereo images. In order to reduce the computational cost of the bias correction procedure, we will establish our technique based on an estimation method referred to as the *sigma-point* method in the literature. The method is based on deterministic sampling of a probability distribution. Through this approach, the deterministic samples are transformed by a nonlinear function and the results are used to estimate the statistics associated with the transformed distribution. The sigma-point method is also called the *unscented transform* and has applications in nonlinear filtering and control systems used for integrated navigation systems [33], spacecraft attitude estimation [22], and underwater navigation [98]. Specifically, we will perform modifications on the sigma-point method to account for the effect of the disparity threshold. In addition, we will present another approach to estimate bias in VO problems based on the bias estimation method proposed by Box [14] in 1971. Box presents a general expression for estimating bias in nonlinear least squares problems. In this thesis, we aim to rederive this expression for the VO setup and utilize it for bias estimation of VO path estimates.

This thesis is structured as follows. Chapter 2 outlines the visual odometry pipeline and shows that our bias estimation and correction algorithm can be added to the end of the pipeline as a new block. It also details the system setup including the

configuration of the landmarks and the camera pose. Furthermore, it describes the stereo camera model in detail and defines the disparity threshold. Chapter 3 covers the steps to produce noisy measurements and the bundle adjustment algorithm for state estimation used in the thesis. Bundle adjustment utilizes the Gauss-Newton algorithm in order to estimate the system state consisting of position of the landmarks and the camera pose. Chapter 4 utilizes a set of symmetric and asymmetric landmark configurations in 3D space to identify VO bias in certain directions. This chapter outlines bias estimation machinery and its modification to account for the effect of the disparity threshold in the stereo-triangulation problem. Simulation results are provided to demonstrate how bias is changed with respect to the variation of parameters such as the disparity threshold, the standard deviation of the system noise, and the position of the landmarks. Chapter 5 begins the study of bias estimation in VO problems by making the simplifying assumption of knowing ground truth. We first consider this assumption to show that bias exists in estimated results even if ground truth is used in position estimation algorithms. This assumption is then removed, and the bootstrap idea is adapted to compute bias estimates based on the measurements and the state estimate. It covers the mathematical formulation for bias correction and provides detailed demonstration of simulation results before and after compensating for bias. Chapter 6 concludes this thesis by summarizing the contributions of our research and discussing areas that could be studied as future work.

Chapter 2

Three-dimensional Scene Reconstruction

2.1 Vision Formation

To understand the advantage of capturing two images of an object from different viewpoints, we refer to the human visual system. The reason that we have two eyes is that this setup provides us with two images of a scene from two distinct view positions at the same time. This gives us information about the objects in three-dimensional space. In order to mimic the human visual system machine vision, we utilize cameras that capture two images of a 3D point concurrently from two different viewpoints. The camera that produces the image pairs is known as the *stereo* camera [23]. A setup of a stereo camera attached to a mobile robot is portrayed in Figure 2.1. In machine visual systems, the depth information is recovered using the stereo-pair images and the known camera poses (translations and orientations) through *triangulation*. The triangulation process results in the estimation of a point's 3D location [91].

Figure 2.1: RO6 robot in ASRL facilities with an attached stereo camera. Photo credit: UTIAS/ASRL [2].

2.2 Visual Odometry Pipeline

Visual Odometry is the use of consecutive captured stereo images to estimate the change in pose of a camera that is used to determine the distance travelled [79]. VO is widely used in the robotics field for the navigation of rovers. An example of VO application is its use on the Mars Exploration Rovers [67]. The captured images are processed to extract the desired information. The architecture of the required image processing in addition to the motion estimation steps is depicted in Figure 2.2.

The standard stereo VO pipeline is briefly described as background review. The stereo images are the inputs to the system. An example is shown in Figure 2.3. Then the steps described in Figure 2.2 are applied to these images.

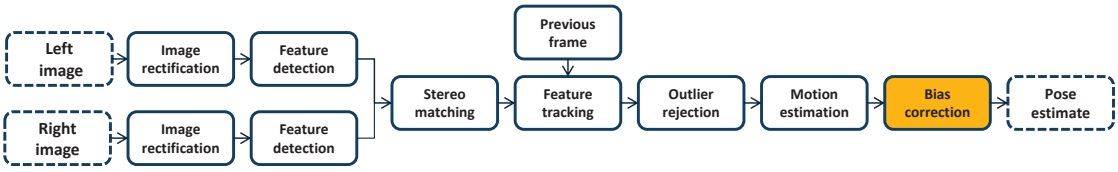

Figure 2.2: Standard stereo visual odometry pipeline with a new block added at the end for bias correction.

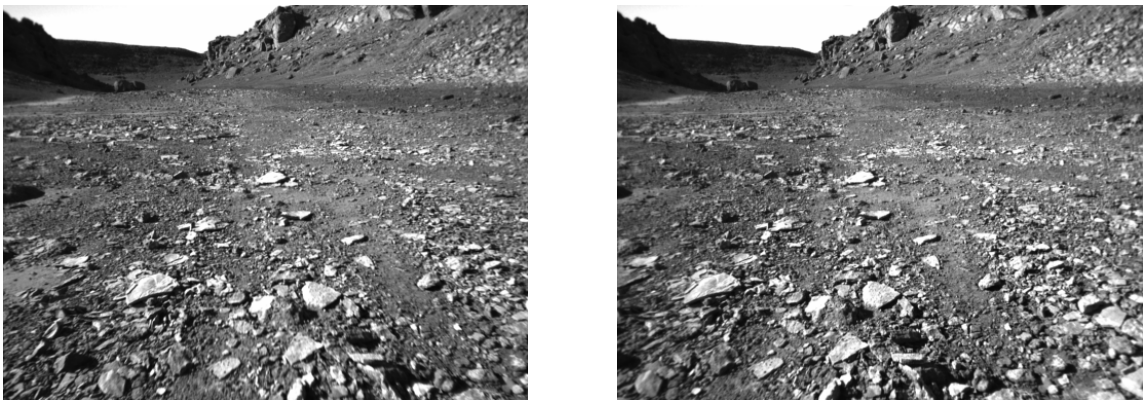

Figure 2.3: Left: left image captured at a Mars analogue site in Devon Island by a stereo camera, right: right image of the stereo pair [41]. Photo credit: ASRL/UTIAS [2].

2.2.1 Image Rectification

The first step in the VO pipeline is the *image rectification* process. After rectifying stereo image pairs, epipolar lines become parallel to the axes of image coordinate frames. Therefore, landmarks are projected onto the same row of the left and right images [47, 4, 36, 40, 84, 94]. This is a desirable feature since it leads to simplified feature matching algorithms. At this step, the effect of lens distortion is also eliminated to refine the quality of the images [23].

2.2.2 Feature Detection

Then the images are prepared for the next step of the VO pipeline, which is *feature detection*. The image features that are detected by feature detectors [86] are also known as *keypoints*. Different techniques exist for detecting keypoints such as the corner detector method [46, 27, 8, 56, 83], the SURF detector mechanism by Bay et al. [7], the Scale Invariant Feature Transform (SIFT) algorithm by Lowe [66], and the FAST (Features from Accelerated Segment Test) feature detector approach by Rosten and Drummond [87, 88]. As an example, the SURF detector technique is employed to detect image features in left and right stereo images and the result is illustrated in Figure 2.4.

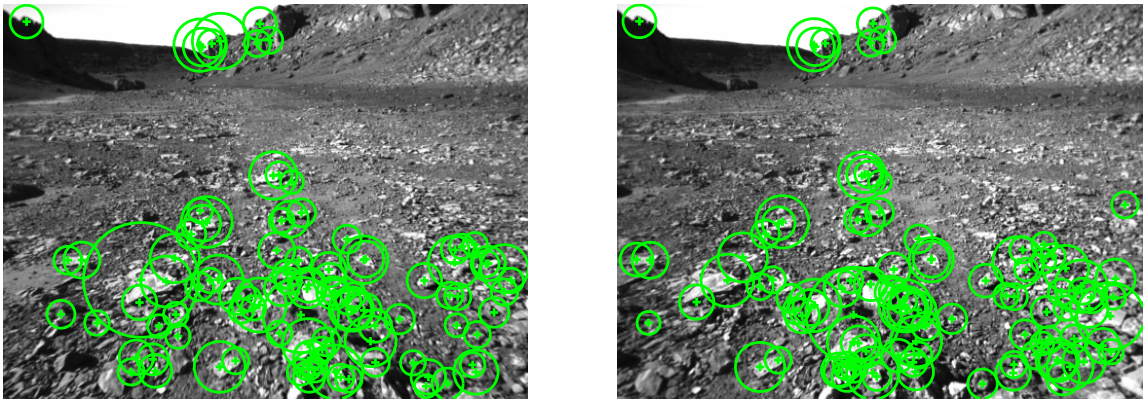

Figure 2.4: The SURF detector is utilized for feature detection in a pair of stereo images. The detected points are shown with green circles in left (left) and right (right) images. The detector provides a set of highly distinctive features that are invariant to image scale and rotation. Furthermore, this approach is faster than the previous feature detectors and has application in camera calibration and object recognition fields [7].

2.2.3 Stereo Matching

From the previous step, image features are detected in left and right stereo images. Using *Stereo Matching*, we find every pair of image features corresponding to a unique landmark. A feature in the left image is matched with the feature in the right image if both represent the same landmark. There are different image matching approaches in the literature including *Area-based Matching* and *Feature-based Matching* algorithms. In the Area-based Matching technique a reference image patch is selected from the left (or right) image and the corresponding one is searched in the right (or left) image using test patches. The test patch in the right (or left) image that has the most correlation with the reference patch is then called the matched patch. This approach is considered in the category of *dense* image matching algorithms since it finds the correspondences of all the image pixels [52] and has 3D scanning applications.

However, sometimes searching for corresponding points of all the image pixels is not required. The process of matching images could be faster and more efficient when only a number of selected distinct features are matched in stereo images. This approach is referred to as the *sparse* image matching algorithm since it provides sparse correspondence of points. The method is used in the standard VO pipeline for the purpose of motion estimation and is known as the *Feature-based Matching* algorithm. A detailed overview providing a comparison of different image matching techniques is given by Brown et al. [17] and Babbar et al. [10].

For the purpose of matching features, a feature can be selected in the left or right image and a search process for finding the corresponding point may be performed by testing either all the pixels in the other image (2D space search) or only a certain set of pixels located on the corresponding epipolar line (1D space search). The former approach is taken by Howard [51] and Konolige et al. [58] and the latter approach is

used by Mei et al. [73] and Nister et al. [80].

After completing the feature matching algorithm, the left and right images are overlaid on top of each other and the coordinates of the matched features are connected. The distance between the matched points is the disparity measurement for those matched features. The disparity lines are drawn in Figure 2.5. Disparity measurements are smaller for the keypoint observations of landmarks that are located farther from the camera.

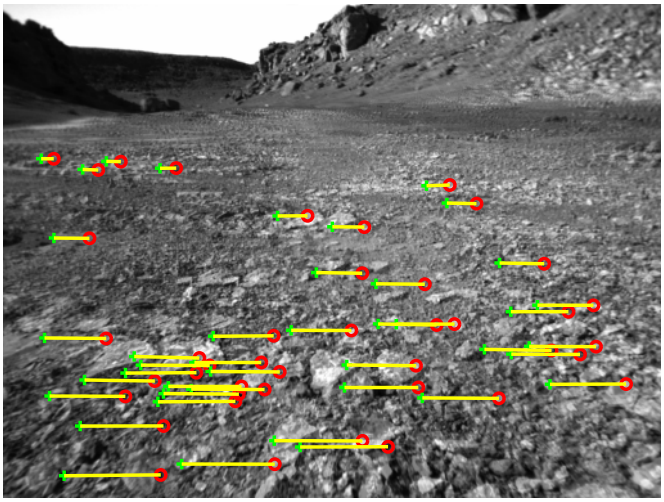

Figure 2.5: The detected features in the left and right images are shown in red and green respectively. The yellow lines show the disparity measurements for the matched features.

It is observed that the disparity value is smaller for the keypoint observations of the landmarks that are located farther from the stereo camera in the 3D environment.

2.2.4 Feature Tracking and Outlier Rejection

The next step in the VO pipeline is to perform another feature matching. This time, however, features are detected and matched between two consecutive stereo image

pairs that are captured by the mobile vehicle at a previous time and the current time. In Section 2.2.3, the search area to find the matched features could be confined to epipolar lines. In this section, the epipolar lines corresponding to images captured at different time frames are not collinear with each other; thus, for feature tracking, the search area is not defined by epipolar lines. To define a search window containing the corresponding feature some information about the vehicle's motion may be obtained by other sensors or using wheel odometry [18] or by assuming that the vehicle moves with a constant velocity [24]. As illustrated in Figure 2.6 (left), we then compute the disparity measurements for pairs of matched features associated with images from two consecutive stereo frames. In addition, it is assumed that the camera moves in such a way that a set of common features is observed in consecutive stereo images.

Suppose $\mathbf{q}_1$ is a point on the first image and $\mathbf{q}_2$ is a point on the second image. Denote the projection of $\mathbf{q}_2$ onto the first image based on a transformation matrix by $\mathbf{q'}_2$. Then, $\mathbf{q}_1$ and $\mathbf{q}_2$ are considered as *inliers* if the measured distance between $\mathbf{q}_1$ and $\mathbf{q'}_2$ falls within a certain threshold. Erroneous matches that are not considered as inliers are referred to as *outliers*. The effect of only one outlier can cause a significant error in the next step which is the rover motion estimation. For this, a technique is sought to improve the output of the feature tracking algorithm by removing the outliers and keeping the inliers.

A general parameter estimation approach which is used for the purpose of outlier removal is called the Random Sample Consensus (RANSAC) coined by Fischler and Bolles [37]. The algorithm starts with a set of samples which are randomly chosen from the measurements. Then a model is fitted to these samples and the number of other points that are in consensus with this model estimate is recorded. The process is repeated for several iterations and the model estimate with the maximal number of satisfying measurements is selected as the best model. At the end, the best model

can be used to specify which measurements are counted as outliers. The RANSAC approach is highly robust and a more comprehensive description of the method is presented by Hartley and Zisserman [47].

The M-estimator SAmple Consensus (MSAC) algorithm which is a variant of the RANSAC algorithm is applied to the tracked features in Figure 2.6 (left) and the output is portrayed in Figure 2.6 (right). The method may remove all the outliers at the cost of eliminating several inliers. However, excluding some of inliers is not troublesome as long as a sufficiently large number of them are present. The RANSAC method is used in the VO literature by Maimone et al. [67] and Konolige et al. [58]. In addition, Mei et al. [73] utilizes the RANSAC algorithm to present a localisation and mapping system.

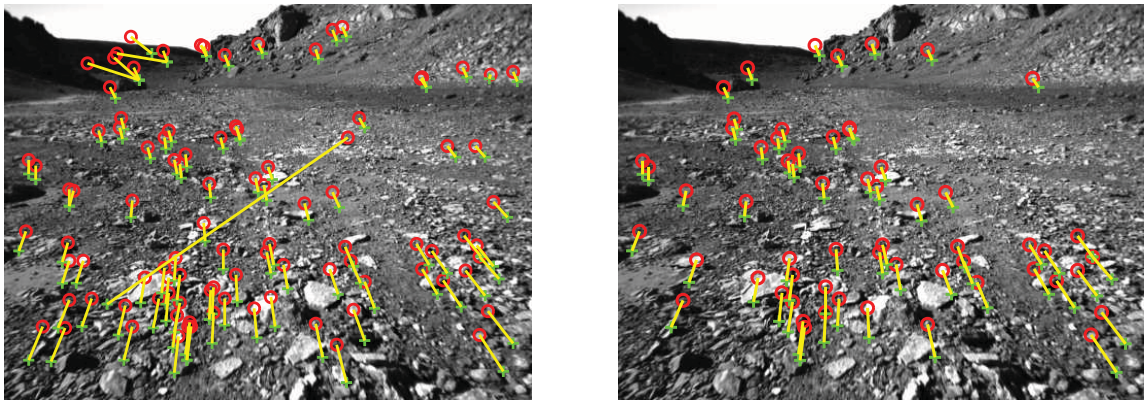

Figure 2.6: The process of feature tracking including outliers (left) and after removing outliers (right) using the GeometricTransformEstimator function from MATLAB. The detected features in the image captured at a previous time are in red and the ones related to the next time step are in green. The yellow lines show the disparity of the matched features taken at two consecutive time steps.

2.2.5 Motion Estimation

The next step after the process of outlier removal is to estimate the rover motion using the feature tracks in the two frames. The rover motion is composed of translation and

orientation of the rover (also called rover pose) from one frame to another. Furthermore, the feature tracks could be employed to estimate the 3D locations of the points; however, these landmark position estimates will not be used in the VO problem. A well-known method for estimating the rover motion is Bundle Adjustment introduced by Brown [15]. In the Bundle Adjustment problem the 3D reconstruction is refined based on an optimality criterion to generate motion and structure estimates. The approach utilizes a minimization technique such as nonlinear least-squares in order to minimize the error between the predicted image points and the observed image points that are corrupted with noise. Bundle Adjustment is a maximum likelihood estimator when the noise is zero-mean Gaussian.

Among the minimization methods for solving nonlinear least-squares problems are the Levenberg-Marquardt method [65, 68, 78] and the Gauss-Newton algorithm [12, 38, 81]. The Bundle Adjustment method with the Gauss-Newton algorithm is used in this thesis and Chapter 3 describes in detail the technique.

2.2.6 Bias Correction

In the standard VO pipeline (refer to Figure 2.2), we capture two images, detect features, stereo match features, temporally track features, reject outlier feature tracks, and finally estimate motion.

Recently, it is reported in some works that when motion estimates are computed for a sufficient number of runs, the average over these runs deviates from ground truth. This error is termed *bias* in VO and is a recent topic of study. In this thesis, we investigate the bias problem and propose to add a new block to the VO pipeline to handle bias estimation and correction. A comprehensive study of bias along with our novel approach to estimate it and use it to improve estimation accuracy is described

in Chapters 4, 5, and 6.

2.3 System Setup

A simplified setup is used in this thesis that is composed of a stereo camera and a collection of 3D points (landmarks) located in the camera's field of view. The elements that describe the motion of the camera in 3D space will be introduced in Subsection 2.3.1. Furthermore, a mathematical model of the camera that will be used to obtain the image coordinates of landmarks is covered in detail in Subsection 2.3.2.

2.3.1 Setup Components

This subsection mainly describes the components used in the system setup that will determine the position of landmarks and the pose of the camera in 3D space. A stereo camera is attached to the front of the rover whose position and orientation are to be estimated. The setup is shown in Figure 2.7; a stereo camera observes some landmarks at time k', moves, then reobserves the landmarks at time k. Thus the basic frame-to-frame VO is performed in this thesis versus the other alternative of multi-frame VO in which camera moves multiple times capturing features at each frame and then matches and tracks features in all the frames.

Figure 2.7: Location of landmarks and camera poses in 3D space.

Before outlining our system model we provide a brief review of the special Euclidean group *SE(3)* which describes the rigid body motion of an object. As illustrated in the scheme of Figure 2.8, a reference frame is fixed in space and the moving frame

which is attached to the object rotates and translates in 3D space (for more details, see [11]). The transformation of the moving frame is then expressed by the 4×4 homogeneous transformation matrix, $\mathbf{T}$, as follows:

$$SE(3) = \left\{ \mathbf{T} | \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix}, \mathbf{C} \in SO(3), \mathbf{d} \in \mathbb{R}^3 \right\}, \tag{2.1}$$

where *SO(3)* is called the rotation group on $\mathbb{R}^3$ [11]. Moreover, we know the inverse of the transformation matrix, $\mathbf{T}$, exists since the rotation matrix, $\mathbf{C}$, is invertible.

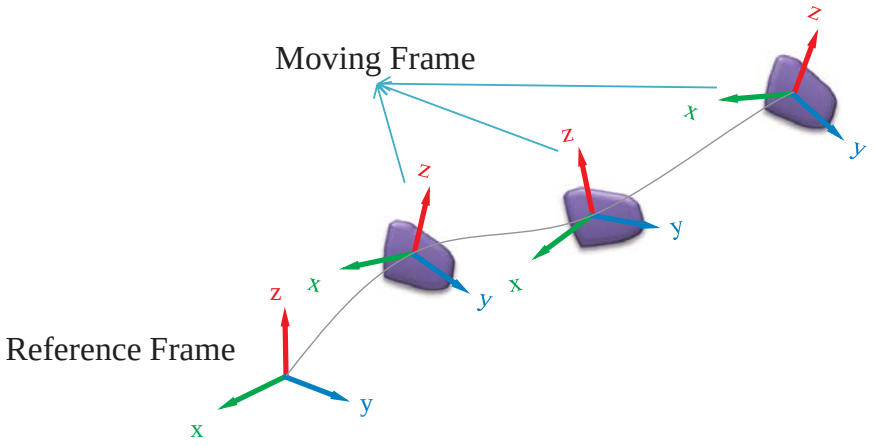

Figure 2.8: Displacements of an object in 3D space with respect to a reference frame.

Our system model is characterized by system parameters that provide information about the position of a landmark and the camera pose consisting of translation and rotation. Let

$$\boldsymbol{\epsilon}_j := \begin{bmatrix} x \\ y \\ z \end{bmatrix}_j. \tag{2.2}$$

Then the 4×1 homogeneous point representing the 3D position of landmark j in homogeneous coordinates [57] is defined as

$$\mathbf{p}_j := \begin{bmatrix} \boldsymbol{\epsilon}_j \\ 1 \end{bmatrix}. \tag{2.3}$$

The pose of the camera at time k can be interpreted using a 4×4 transformation matrix, $\mathbf{T}_k$, or a 6×1 camera pose vector, $\boldsymbol{\pi}_k$. We use both $\mathbf{T}_k$ and $\boldsymbol{\pi}_k$ to present the camera pose in the following chapters. Depending on which quantity is obtained from equations and which quantity is desired, we can convert one to the other. To this end, we describe the elements forming $\mathbf{T}_k$ and $\boldsymbol{\pi}_k$.

Interpreted in the base frame, the translation of the current (time k) camera frame from the base frame is described by a 3×1 translation vector denoted $\mathbf{r}_k$. The rotation of the current camera frame (time k) about a unit-length axis $\mathbf{a}_k := (a_x, a_y, a_z)_k$ with an angle ϕ_k can be expressed using the 3×1 camera rotation vector $\boldsymbol{\phi}_k$ [49] defined as

$$\boldsymbol{\phi}_k := \phi_k \, \mathbf{a}_k, \qquad \mathbf{a}_k \in \mathbb{R}^3. \tag{2.4}$$

The cross-product operation $(\cdot)^\times$ [26] in *SO(3)* that transforms a 3×1 vector to a 3×3 matrix is defined by

$$\mathbf{w}^\times := \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}^\times := \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix}. \tag{2.5}$$

Using Rodrigues' rotation formula [74], the 3×3 camera rotation matrix represented as $\mathbf{C}_k$ can be stated as

$$\begin{aligned} \mathbf{C}_k &:= e^{-{\boldsymbol{\phi}_k}^\times} \\ &= \sum_{n=0}^{\infty} (-1)^n \frac{1}{n!} ({\boldsymbol{\phi}_k}^\times)^n \\ &= \cos\phi_k \mathbf{1} + (1 - \cos\phi_k)\mathbf{a}_k {\mathbf{a}_k}^T - \sin\phi_k {\mathbf{a}_k}^\times. \end{aligned} \tag{2.6}$$

Define $\mathbf{S}_k$ as

$$\begin{aligned} \mathbf{S}_k &:= \int_0^1 (\mathbf{C}_k)^\alpha d\alpha \\ &= \sum_{n=0}^{\infty} (-1)^n \frac{1}{(n+1)!} (\boldsymbol{\phi}_k{}^\times)^n \\ &= \frac{\sin\phi_k}{\phi_k}\mathbf{1} + \left(1 - \frac{\sin\phi_k}{\phi_k}\right)\mathbf{a}_k\mathbf{a}_k{}^T - \frac{1-\cos\phi_k}{\phi_k}\mathbf{a}_k{}^\times, \end{aligned} \tag{2.7}$$

and let $\boldsymbol{\rho}_k$, which is a 3×1 vector, be the solution to the equation

$$\mathbf{S}_k\boldsymbol{\rho}_k = -\mathbf{C}_k\,\mathbf{r}_k. \tag{2.8}$$

These elements form a 4×4 transformation matrix which determines the pose of the camera at time k in the base frame (initial frame) as follows:

$$\mathbf{T}_k := \begin{bmatrix} \mathbf{C}_k & -\mathbf{C}_k\,\mathbf{r}_k \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_k & \mathbf{S}_k\boldsymbol{\rho}_k \\ \mathbf{0}^T & 1 \end{bmatrix}, \qquad \mathbf{T}_k \in \mathbb{R}^{4\times 4} \tag{2.9}$$

For more details on the construction of the transformation matrix $\mathbf{T}$ using the rotation and translation quantities, refer to [74]. The camera pose is also stated in a 6×1 vector denoted by $\boldsymbol{\pi}_k$ as follows:

$$\boldsymbol{\pi}_k = \begin{bmatrix} \boldsymbol{\rho}_k \\ \boldsymbol{\phi}_k \end{bmatrix}, \qquad \boldsymbol{\pi}_k \in \mathbb{R}^6 \tag{2.10}$$

Define the *SE(3)* operator $(\cdot)^\boxplus$ that transforms a 6×1 vector to a 4×4 matrix by

$$\mathbf{w}^\boxplus = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}^\boxplus := \begin{bmatrix} \mathbf{v}^\times & -\mathbf{u} \\ \mathbf{0}^T & 0 \end{bmatrix}. \tag{2.11}$$

where $\mathbf{u}$ and $\mathbf{v}$ are 3×1 vectors. The relationship between $\mathbf{T}_k$ and $\boldsymbol{\pi}_k$ is given by

$$\mathbf{T}_k = e^{-\boldsymbol{\pi}_k^\boxplus}. \tag{2.12}$$

To prove (2.12) we start from the right hand side of the equation:

$$\begin{aligned} e^{-\boldsymbol{\pi}_k^{\boxplus}} &= e^{-\begin{bmatrix}\boldsymbol{\rho}_k \\ \boldsymbol{\phi}_k\end{bmatrix}^{\boxplus}} \\ &= e^{-\begin{bmatrix}\boldsymbol{\phi}_k^{\times} & -\boldsymbol{\rho}_k \\ \mathbf{0}^T & 0\end{bmatrix}} \\ &= \sum_{n=0}^{\infty} \frac{1}{n!}\left(-\begin{bmatrix}\boldsymbol{\phi}_k^{\times} & -\boldsymbol{\rho}_k \\ \mathbf{0}^T & 0\end{bmatrix}\right)^n \\ &= \mathbf{1} - \begin{bmatrix}\boldsymbol{\phi}_k^{\times} & -\boldsymbol{\rho}_k \\ \mathbf{0}^T & 0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\boldsymbol{\phi}_k^{\times} & -\boldsymbol{\rho}_k \\ \mathbf{0}^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\phi}_k^{\times} & -\boldsymbol{\rho}_k \\ \mathbf{0}^T & 0\end{bmatrix} + \cdots \end{aligned} \tag{2.13}$$

Then we sum the above terms and use (2.6) and (2.7) to obtain

$$\begin{bmatrix} 1 - \boldsymbol{\phi}_k^{\times} + \frac{1}{2}(\boldsymbol{\phi}_k^{\times})^2 - \cdots & (1 - \frac{1}{2}\boldsymbol{\phi}_k^{\times} + \cdots)\boldsymbol{\rho}_k \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_k & \mathbf{S}_k\boldsymbol{\rho}_k \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{2.14}$$

The right hand side of (2.14) is the transformation matrix $\mathbf{T}_k$. Furthermore, we can define the state vector, $\mathbf{x}_k$, at time k, which determines the camera pose and the position of J landmarks at the current time:

$$\mathbf{x}_k = \begin{bmatrix} \boldsymbol{\pi}_k \\ \boldsymbol{\epsilon}_1 \\ \vdots \\ \boldsymbol{\epsilon}_J \end{bmatrix}, \qquad \boldsymbol{\pi}_k \in \mathbb{R}^6, \quad \mathbf{x}_k \in \mathbb{R}^{(6+3J)} \tag{2.15}$$

where $\boldsymbol{\epsilon}_j$ and $\boldsymbol{\pi}_k$ are given in (2.2) and (2.10), respectively. The quantities introduced in this subsection will be used in the subsequent chapters to formulate the bias estimation algorithm in VO problems.

2.3.2 Stereo Camera Model

The nonlinear measurement function in the setup (i.e., the stereo camera model) with its corresponding parameters is portrayed in Figure 2.9.

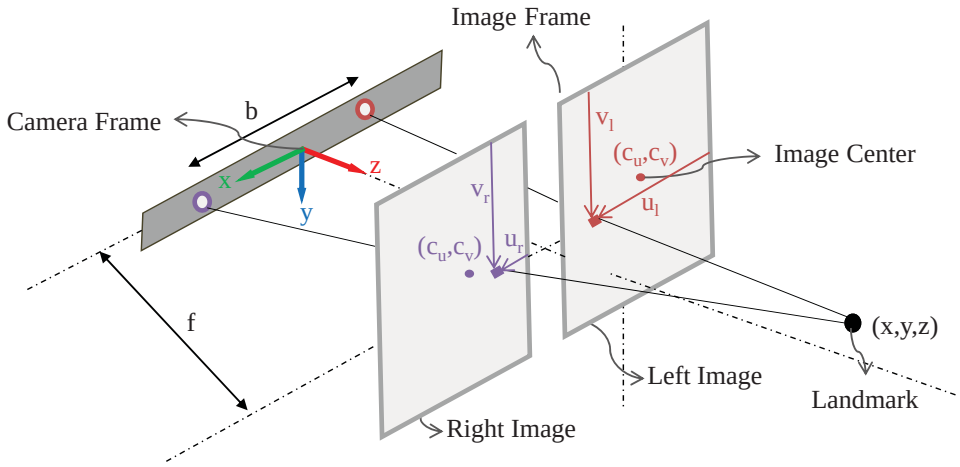

Figure 2.9: One landmark is projected into the left and right images of a stereo camera. The stereo camera model parameters are also shown above.

The stereo camera model, $\mathbf{f}(\cdot)$, is used to compute the 4×1 image coordinates, $\mathbf{y}_{jk}$, of each landmark at time k. The position of landmark j in the current camera frame is given by

$$\mathbf{g}_{jk} := \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ 1 \end{bmatrix}_{jk} = \mathbf{T}_k \mathbf{p}_j. \tag{2.16}$$

The model is presented by the following equation

$$\mathbf{y}_{jk} := \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix}_{jk} = \begin{bmatrix} \frac{f_u}{g_3}(g_1 + \frac{b}{2}) \\ \frac{f_v}{g_3} g_2 \\ \frac{f_u}{g_3}(g_1 - \frac{b}{2}) \\ \frac{f_v}{g_3} g_2 \end{bmatrix}_{jk} + \begin{bmatrix} c_u \\ c_v \\ c_u \\ c_v \end{bmatrix}, \quad \begin{matrix} f_u = s_u f \\ f_v = s_v f \end{matrix}, \tag{2.17}$$

where $\{s_u, s_v\}$ are the horizontal and vertical pixel conversions in pixels/metre, f and

b are the focal length and the camera baseline in metres, and (c_u, c_v) is the center of the image plane in pixels. Equivalently, the camera model denoted by $\mathbf{f}(\cdot)$ is expressed as follows:

$$\mathbf{f}(\mathbf{g}_{jk}) = \mathbf{M}\frac{1}{g_{3,jk}}\,\mathbf{g}_{jk}, \tag{2.18}$$

where $\mathbf{M}$ is the camera matrix and contains all the parameters associated with the camera such as focal length, image center, and baseline

$$\mathbf{M} := \begin{bmatrix} f_u & 0 & c_u & f_u\frac{b}{2} \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u\frac{b}{2} \\ 0 & f_v & c_v & 0 \end{bmatrix}. \tag{2.19}$$

Using (2.16) and (2.17), the inverse of the camera model, $\mathbf{f}^{-1}(\cdot)$, is given by

$$\mathbf{f}^{-1}(\mathbf{y}_{jk}) = \mathbf{g}_{jk} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ 1 \end{bmatrix}, \quad \begin{cases} g_1 = \frac{u_l + u_r - 2c_u}{2f_u} g_3, \\ g_2 = \frac{v_l - c_v}{f_v} g_3, \\ g_3 = \frac{f_u b}{u_l - u_r},\ u_l \neq u_r. \end{cases} \tag{2.20}$$

2.3.3 Disparity Threshold

The distance between the horizontal pixel coordinates of the matched points of a stereo image pair is referred to as the *horizontal disparity* (or simply *disparity*[1]).

[1]The term *disparity* is a common term used in the stereo matching algorithms in the three-dimensional data reconstruction literature. In the algorithms, measurements with disparities less than a threshold value are usually erroneous measurements and are discarded from the set of observations.

Using the above parameters the disparity, d, of a landmark observation is defined as

$$d = u_l - u_r. \tag{2.21}$$

As depicted in Figure 2.9, $u_l > u_r$ (i.e. the disparity is positive) for a landmark located in front of the camera. Furthermore, the disparity becomes small for far-away landmarks. A *disparity threshold*, d_{th}, is considered in our analysis to truncate undesired measurements. It is assumed that the camera is not capable of accurately capturing far landmarks or landmarks behind it. Hence, the measurements that have disparities less than d_{th} are truncated (i.e., discarded).Typically a small positive constant is used for d_{th}.

It is shown in subsequent chapters that applying the disparity threshold to measurements is one source of bias in position estimation problems. The more samples are truncated in our problem, the larger the bias is observed in the result.

Chapter 3

State Estimation Algorithm

3.1 Background

3.1.1 Least Squares Problems

In least squares problems, an objective function is obtained by the sum of some squared terms that determine the discrepancies between measurements and predictions. These terms are referred to as residuals $\mathbf{a}_l$ and are given by

$$\mathbf{a}_l(\hat{\boldsymbol{\alpha}}) = \boldsymbol{\beta}_l - \boldsymbol{\psi}_l(\hat{\boldsymbol{\alpha}}), \qquad \text{for} \qquad l = 1...L \tag{3.1}$$

where $\boldsymbol{\beta}_l$ are measurements and $\boldsymbol{\psi}_l(\hat{\boldsymbol{\alpha}})$ are predictions. Then, the $\mathbf{a}_l(\hat{\boldsymbol{\alpha}})$'s which are functions from $\mathbb{R}^n$ to $\mathbb{R}^m$ $(L\,m \geq n)$ define the least squares objective function $Q(\hat{\boldsymbol{\alpha}})$ as follows [81]:

$$Q(\hat{\boldsymbol{\alpha}}) = \frac{1}{2}\sum_{l=1}^{L} \|\mathbf{a}_l(\hat{\boldsymbol{\alpha}})\|^2 \tag{3.2}$$

where $\|\cdot\|$ represents the Euclidean norm. In least squares problems, the objective function $Q(\hat{\boldsymbol{\alpha}})$ is minimized to solve for the parameter vector $\hat{\boldsymbol{\alpha}}$. A least squares

problem is categorized as linear or nonlinear based respectively on the linearity or nonlinearity of residual functions $\mathbf{a}_l = (a_{l,1}, a_{l,2}, ..., a_{l,m})^T$.

3.1.2 Newton's Method

While there is a wide variety of methods to optimize an objective function, we focus here on iterative algorithms. These algorithms use an initial guess as their starting point denoted by $\hat{\boldsymbol{\alpha}}(0)$ and seek to improve it in each iteration until they converge to an optimal point represented as $\hat{\boldsymbol{\alpha}}^\star$. The current estimate $\hat{\boldsymbol{\alpha}}(i)$ and the improved estimate $\hat{\boldsymbol{\alpha}}(i+1)$ are linked by

$$\hat{\boldsymbol{\alpha}}(i+1) = \hat{\boldsymbol{\alpha}}(i) + \delta\hat{\boldsymbol{\alpha}}(i), \tag{3.3}$$

where $\delta\hat{\boldsymbol{\alpha}}(i)$ is the improvement term. There are different methodologies for obtaining an improved estimate in each iteration. Using (3.1) and (3.2), define the gradient $\nabla Q(\hat{\boldsymbol{\alpha}})$ and the Hessian $\boldsymbol{\mathcal{H}}_Q(\hat{\boldsymbol{\alpha}})$ as follows [81]:

$$\nabla Q(\hat{\boldsymbol{\alpha}}) = \sum_{l=1}^{L} \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}})^T \mathbf{a}_l(\hat{\boldsymbol{\alpha}}), \tag{3.4}$$

$$\boldsymbol{\mathcal{H}}_Q(\hat{\boldsymbol{\alpha}}) = \sum_{l=1}^{L} \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}})^T \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}}) + \sum_{l=1}^{L}\sum_{i=1}^{m} a_{l,i}(\hat{\boldsymbol{\alpha}})\boldsymbol{\mathcal{H}}_{a_{l,i}}(\hat{\boldsymbol{\alpha}}), \tag{3.5}$$

where ∇ is the gradient operator and $\mathbf{H}_{\mathbf{a}_l}$ is the Jacobian of $\mathbf{a}_l$. In Newton's method, $\delta\hat{\boldsymbol{\alpha}}(i)$ is obtained from

$$-\boldsymbol{\mathcal{H}}_Q(i)\delta\hat{\boldsymbol{\alpha}}(i) = \nabla Q(i), \tag{3.6}$$

supposing that the objective function $Q(\hat{\boldsymbol{\alpha}})$ is twice differentiable and its Hessian $\boldsymbol{\mathcal{H}}_Q(\hat{\boldsymbol{\alpha}})$ is Lipschitz continuous. It can be shown [81] that for a starting point $\hat{\boldsymbol{\alpha}}(0)$ sufficiently close to $\hat{\boldsymbol{\alpha}}^\star$, the iterative algorithm converges quadratically to $\hat{\boldsymbol{\alpha}}^\star$.

3.1.3 Gauss-Newton Method

In some least squares problems, the first term in (3.5) dominates the second term when either the residuals $\mathbf{a}_l(\hat{\boldsymbol{\alpha}})$ or the residual Hessians $\boldsymbol{\mathcal{H}}_{a_{l,i}}(\hat{\boldsymbol{\alpha}})$ are relatively small. This will make problems analytically and computationally efficient; however, to ignore the second term in (3.5) which is difficult to compute, $\hat{\boldsymbol{\alpha}}$ may need to be sufficiently close to the optimal solution $\hat{\boldsymbol{\alpha}}^{\star}$ [81]. The Gauss-Newton method which is a modification to Newton's method is a solution to nonlinear least squares problems and is based on approximating (3.5) as follows:

$$\boldsymbol{\mathcal{H}}_Q(\hat{\boldsymbol{\alpha}}) \approx \sum_{l=1}^{L} \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}})^T \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}}), \tag{3.7}$$

When $\sum_{l=1}^{L} \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}})^T \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}})$ is a close approximation to $\boldsymbol{\mathcal{H}}_Q(\hat{\boldsymbol{\alpha}})$, the rate of convergence for the Gauss-Newton algorithm becomes the same as that for Newton's algorithm (i.e. the number of iterations until convergence is the same for both algorithms). Furthermore, the Gauss-Newton method is computationally less expensive than Newton's method in each iteration [81, 82, 95]. Therefore, the Gauss-Newton algorithm performs faster than the Newton's method. As a drawback of Newton's algorithms, $\delta\hat{\boldsymbol{\alpha}}$ may not always be a descent direction, since the Hessian matrix $\boldsymbol{\mathcal{H}}_Q(\hat{\boldsymbol{\alpha}})$ may not always be positive definite whereas, in the Gauss-Newton method, this condition will be omitted through the approximation of the Hessian matrix with the positive semidefinite expression (3.7). For more details on the advantages of the Gauss-Newton method over the Newton's method read the Numerical Optimization textbook by Nocedal and Wright [81].

Using Taylor's theorem, the Gauss-Newton algorithm can be performed by linearizing each residual $\mathbf{a}_l(\hat{\boldsymbol{\alpha}})$ in (3.1) around a nominal vector $\bar{\boldsymbol{\alpha}}$. The linear approximation of $\mathbf{a}_l(\hat{\boldsymbol{\alpha}})$ using a first-order Taylor series expansion is given by

$$\begin{aligned} \mathbf{a}_l(\bar{\boldsymbol{\alpha}} + \delta\hat{\boldsymbol{\alpha}}) &= \boldsymbol{\beta}_l - \boldsymbol{\psi}_l(\bar{\boldsymbol{\alpha}} + \delta\hat{\boldsymbol{\alpha}}) \\ &\approx \boldsymbol{\beta}_l - \boldsymbol{\psi}_l(\bar{\boldsymbol{\alpha}}) - \mathbf{H}_{\boldsymbol{\psi}_l}(\bar{\boldsymbol{\alpha}})^T \, \delta\hat{\boldsymbol{\alpha}}, \end{aligned} \tag{3.8}$$

where $\mathbf{H}_{\boldsymbol{\psi}_l}$ is the Jacobian of $\boldsymbol{\psi}_l$.The nonlinear least squares problem is converted to a linear least squares problem by substituting the linearized residuals in (3.2). The objective function is approximated as:

$$\begin{aligned} Q(\bar{\boldsymbol{\alpha}} + \delta\hat{\boldsymbol{\alpha}}) &= \frac{1}{2}\sum_{l=1}^{m} \|\mathbf{a}_l(\bar{\boldsymbol{\alpha}} + \delta\hat{\boldsymbol{\alpha}})\|^2 \\ &\approx \frac{1}{2}\sum_{l=1}^{m} \|(\boldsymbol{\beta}_l - \boldsymbol{\psi}_l(\bar{\boldsymbol{\alpha}}) - \mathbf{H}_{\boldsymbol{\psi}_l}(\bar{\boldsymbol{\alpha}})^T \, \delta\hat{\boldsymbol{\alpha}})\|^2. \end{aligned} \tag{3.9}$$

In order to compute the minimizer of the objective function (3.9), we use the initial minimizer $\hat{\boldsymbol{\alpha}}(0)$ such that:

$$-\left(\sum_{l=1}^{L} \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}}(0))^T \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}}(0))\right) \delta\hat{\boldsymbol{\alpha}}(0) = \sum_{l=1}^{L} \mathbf{H}_{\mathbf{a}_l}(\hat{\boldsymbol{\alpha}}(0))^T \, (\mathbf{y}_l - \mathbf{f}_l(\hat{\boldsymbol{\alpha}}(0))) \,. \tag{3.10}$$

then updating by $\hat{\boldsymbol{\alpha}}(1) = \hat{\boldsymbol{\alpha}}(0) + \delta\hat{\boldsymbol{\alpha}}(0)$ gives a new initial point. We keep iterating and updating this step until the desired result obtained, that is, $\|\delta\hat{\boldsymbol{\alpha}}(i) - \delta\hat{\boldsymbol{\alpha}}(i-1)\| \leq \varepsilon$ for some small positive value of given ε. This process will guarantee [44, 82] that the Gauss-Newton iteration converges locally to $\hat{\boldsymbol{\alpha}}^\star$.

3.2 Bundle Adjustment for Mobile Robot State Estimation

Bundle Adjustment (BA) is an estimation problem which utilizes image projections of 3D points to reconstruct some 3D scene and to estimate motion parameters such as camera poses. BA which is formulated as a nonlinear least squares problem was

first used in photogrammetry and has been recently applied to a number of computer vision problems [93, 16, 43, 97]. BA is flexible in choosing the type of camera, landmarks and error models [93]. In Bundle Adjustment, a cost function which is the sum of quadratic error terms is minimized to achieve an optimal array of parameter (landmarks positions and camera poses) estimates. The numerical optimization of the cost function can be performed using different algorithms including the iterative Gauss-Newton algorithm that will be used in this thesis.

It was mentioned in Chapter 2 that the landmark positions in 3D are interpreted in homogeneous coordinates. In the Gauss-Newton algorithm a system of linear equations are solved to find the optimal solution which minimizes the cost function. During this iterative process, problems may arise with the convergence of the algorithm if the far points are stated in Euclidean coordinates as they tend to infinity. Nonetheless, the homogeneous points can easily represent points at infinity and therefore they are beneficial to the Gauss-Newton algorithm [49]. In order to interpret these homogeneous points in the camera frame, a transformation matrix is defined that combines the rotation and translation information associated with the camera frame at each time step.

In the frame-to-frame BA problem, it is supposed that the camera pose is known at a previous time k' and the goal is to estimate the camera pose at the current time k with respect to the previous camera frame. Therefore, the transformation matrix $\hat{\mathbf{T}}_k$ is an estimate which describes the change in the orientation and translation of the camera with respect to the previous camera frame (see Figure 2.8). In this problem, the homogeneous form of the position estimate of landmark j interpreted in the previous camera frame is described by $\hat{\mathbf{p}}_j$. The position estimate of landmark j can be interpreted in the current camera frame using the transformation matrix $\hat{\mathbf{T}}_k$.

Define $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_{jk}$

$$\hat{\mathbf{x}}_k := \begin{bmatrix} \hat{\boldsymbol{\pi}}_k \\ \hat{\boldsymbol{\epsilon}}_1 \\ \vdots \\ \hat{\boldsymbol{\epsilon}}_J \end{bmatrix}, \qquad \hat{\mathbf{x}}_{jk} := \begin{bmatrix} \hat{\boldsymbol{\pi}}_k \\ \hat{\boldsymbol{\epsilon}}_j \end{bmatrix}, \tag{3.11}$$

where $\hat{\boldsymbol{\pi}}_k$ is the estimate of the 6×1 camera pose vector at time k with respect to the previous camera frame and $\hat{\boldsymbol{\epsilon}}_j$ is the estimate of the 3×1 position vector associated with landmark j interpreted in the previous camera frame. The homogeneous form of the landmark position in the current camera frame is obtained by $\mathbf{g}_{jk}$ as follows:

$$\mathbf{g}_{jk}(\hat{\mathbf{x}}_{jk}) = \hat{\mathbf{T}}_k \, \hat{\mathbf{p}}_j, \quad \mathbf{g}_{jk} \in \mathbb{R}^4. \tag{3.12}$$

Expressions (2.3) to (2.10) show the dependence of $\hat{\mathbf{T}}_k$ on $\hat{\boldsymbol{\pi}}_k$ and the dependence of $\hat{\mathbf{p}}_j$ on $\hat{\boldsymbol{\epsilon}}_j$. Therefore, $\mathbf{g}_{jk}$ in (3.12) depends on $\hat{\mathbf{x}}_{jk}$ using (3.11).

The function $\mathbf{h}_{jk}(\cdot)$ determines the predicted image coordinates of landmark j captured by the camera at the current time k. Using (3.12) and the camera model $\mathbf{f}(\cdot)$ from (2.18), we have the following expression

$$\mathbf{h}_{jk}(\hat{\mathbf{x}}_{jk}) = \mathbf{f}(\hat{\mathbf{T}}_k \, \hat{\mathbf{p}_j}). \tag{3.13}$$

The function $\mathbf{h}_{jk'}(\cdot)$ determines the predicted image coordinates of landmark j captured by the camera at a previous time k' using the camera model $\mathbf{f}(\cdot)$, which is described in (2.18), as follows:

$$\mathbf{h}_{jk'}(\hat{\mathbf{x}}_{jk'}) = \mathbf{f}(\hat{\mathbf{p}}_j), \tag{3.14}$$

where

$$\hat{\mathbf{x}}_{jk'} := \begin{bmatrix} \hat{\boldsymbol{\pi}}_{k'} \\ \hat{\boldsymbol{\epsilon}}_j \end{bmatrix}, \quad \hat{\boldsymbol{\pi}}_{k'} = \mathbf{0}_{6\times 1}. \tag{3.15}$$

3.2.1 Simulation Setup

This step covers the process of producing noisy measurements for simulation analysis only. In real VO experiments, acquired images are processed and the measurements for motion estimation are obtained from the output of the outlier rejection block in the VO pipeline described in Chapter 2.

Define the true values of the position vector associated with landmark j and the transformation matrix at time k by $\mathbf{p}_{j,\mathrm{t}}$ and $\mathbf{T}_{k,\mathrm{t}}$, respectively. Let $\mathbf{y}_{jk,\mathrm{t}} = \mathbf{f}(\mathbf{T}_{k,\mathrm{t}}\,\mathbf{p}_{j,\mathrm{t}})$ denote the noise-free pixel coordinates of landmark j at time k. To simulate real measurements, noise with a zero-mean Gaussian distribution $\mathbf{n}_{jk}$ is added to $\mathbf{y}_{jk,\mathrm{t}}$. This results in the corresponding noisy measurements, $\mathbf{y}_{jk}$ composed of four pixel coordinates from left and right images, and is given by

$$\begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix}_{jk} = \mathbf{y}_{jk} = \mathbf{y}_{jk,\mathrm{t}} + \mathbf{n}_{jk}. \tag{3.16}$$

The covariance of the zero-mean Gaussian noise vector $\mathbf{n}_{jk}$ is given by

$$\mathbf{R}_{jk} := \begin{bmatrix} \sigma_{u_l}^2 & & & 0 \\ & \sigma_{v_l}^2 & & \\ & & \sigma_{u_r}^2 & \\ 0 & & & \sigma_{v_r}^2 \end{bmatrix}_{jk}. \tag{3.17}$$

The parameter $\sigma_{u_l}^2$ is the variance of the noise added to the horizontal coordinate of the left image and a similar definition holds for each of the other image coordinates.

In addition, measurements are generated at a previous time k' using $\mathbf{y}_{jk',\mathrm{t}} = \mathbf{f}(\mathbf{p}_{j,\mathrm{t}})$

$$\mathbf{y}_{jk'} = \mathbf{y}_{jk',\mathrm{t}} + \mathbf{n}_{jk'}. \tag{3.18}$$

It is worthwhile to point out that no matter whether the measurements are obtained from real processed data captured by the stereo camera or generated artificially from equation (3.16), it will be referred to as $\mathbf{y}_{jk,\mathrm{meas}}$ and $\mathbf{y}_{jk',\mathrm{meas}}$ (associated with landmark j at time k and time k').

The noisy measurements generated in this subsection will be used to form the residual terms of the objective function of our estimation problem.

3.2.2 Minimization Problem

The Bundle Adjustment method which is based on the Gauss-Newton algorithm is used to estimate the location of the landmarks and the camera poses. The Gauss-Newton algorithm is applied to minimize the objective function Q_{ba} which is constructed by the set of reprojection error vectors labelled $\{\mathbf{e}_{jk}, \mathbf{e}_{jk'}\}$ and to solve for the optimal state vector $\hat{\mathbf{x}}_k^\star$ that includes information about the position of landmarks and the camera pose. Minimizing the objective function provides us with the maximum likelihood estimate for the new camera pose and landmark positions.

To construct the objective function of the system, the error terms are formed. The reprojection error terms represent the difference between the predicted image coordinates of landmarks (obtained using the camera model) and the actual measurements. In VO experiments, these are the real images of the scene captured by the stereo camera moving in a terrain. In simulations these are the noisy measurements generated by our simulator. The definition of the set of error terms at previous and current

times for landmark j is given by

$$\mathbf{e}_{jk}(\hat{\mathbf{x}}_{jk}) := \mathbf{y}_{jk} - \mathbf{h}_{jk'}(\hat{\mathbf{x}}_{jk}), \tag{3.19}$$

$$\mathbf{e}_{jk'}(\hat{\mathbf{x}}_{jk'}) := \mathbf{y}_{jk'} - \mathbf{h}_{jk'}(\hat{\mathbf{x}}_{jk'}). \tag{3.20}$$

The error terms are used to create the system objective function in the BA problem denoted by Q_{ba}:

$$Q_{\text{ba}}(\hat{\mathbf{x}}_k) := \frac{1}{2}\sum_j \left(\mathbf{e}_{jk}(\hat{\mathbf{x}}_{jk})^T \mathbf{R}_{jk}^{-1} \mathbf{e}_{jk}(\hat{\mathbf{x}}_{jk}) + \mathbf{e}_{jk'}(\hat{\mathbf{x}}_{jk'})^T \mathbf{R}_{jk'}^{-1} \mathbf{e}_{jk'}(\hat{\mathbf{x}}_{jk'}) \right), \tag{3.21}$$

where k and k' are fixed. A nonlinear least squares problem is now formulated by the objective function $Q_{\text{ba}}(\hat{\mathbf{x}}_k)$. In this problem, we seek to find $\hat{\mathbf{x}}_k^\star$ that minimizes the objective function.

3.2.3 Gauss-Newton Method

The Gauss-Newton algorithm is an iterative algorithm that uses a set of initial guesses to solve the problem in the first iteration. Then the initial guesses get updated to be used in the next iteration and the algorithm is iterated until convergence. According to Subsection 3.1.3, the nonlinear error terms in (3.21) are to be substituted by their linear approximations and the problem will be solved as a linear least squares problem. To linearize the error terms around a nominal state vector, $\bar{\mathbf{x}}$, we first explain the perturbation from the nominal state, $\delta\hat{\mathbf{x}}$, in the following.

Perturbations

Suppose the nominal state vector $\bar{\mathbf{x}}_{jk}$ is perturbed to

$$\hat{\mathbf{x}}_{jk} := \bar{\mathbf{x}}_{jk} + \delta\hat{\mathbf{x}}_{jk} = \bar{\mathbf{x}}_{jk} + \begin{bmatrix} \delta\hat{\boldsymbol{\pi}}_k \\ \delta\hat{\boldsymbol{\epsilon}}_j \end{bmatrix}. \tag{3.22}$$

The state vector perturbation $\delta\hat{\mathbf{x}}_{jk}$ can be described by $\{\delta\hat{\mathbf{T}}_k, \delta\hat{\mathbf{p}}_j\}$ that determine the perturbations associated with the transformation matrix $\hat{\mathbf{T}}_k$ and the landmark position vector $\hat{\mathbf{p}}_j$. For the *SO(3)* operators $(\cdot)^\times$ and $(\cdot)^\times$ refer to (2.17) and (2.11) (Chapter 2). The perturbation of $\hat{\mathbf{T}}_k$ is approximated by considering up to the first-derivative term in the Taylor series expansion of the corresponding exponential function around zero [74]:

$$\begin{aligned} \hat{\mathbf{T}}_k := \delta\hat{\mathbf{T}}_k\bar{\mathbf{T}}_k &= e^{-\delta\hat{\boldsymbol{\pi}}_k^{\boxplus}}\bar{\mathbf{T}}_k \\ &\approx \left(\mathbf{1} - \delta\hat{\boldsymbol{\pi}}_k^{\boxplus}\right)\bar{\mathbf{T}}_k. \end{aligned} \tag{3.23}$$

The perturbation of $\hat{\mathbf{p}}_j$ is given by

$$\hat{\mathbf{p}}_j := \bar{\mathbf{p}}_j + \delta\hat{\mathbf{p}}_j = \bar{\mathbf{p}}_j + \mathbf{G}_{jk'}\,\delta\hat{\mathbf{x}}_{jk'}. \tag{3.24}$$

where

$$\mathbf{G}_{jk'} := \begin{bmatrix} \mathbf{0}_{4\times 6} & \mathbf{P} \end{bmatrix}, \quad \text{and} \quad \mathbf{P} := \begin{bmatrix} \mathbf{1}_{3\times 3} \\ \mathbf{0}_{1\times 3} \end{bmatrix} \tag{3.25}$$

is a projection matrix to transform 3×1 landmark perturbation vectors to 4×1 homogeneous vectors.

Linearization

This system contains two nonlinearities that will be linearized in the development of the estimation algorithm. The first is the transformation of landmarks into the camera frame at current time k (i.e. when camera moves to the next frame in the

frame-to-frame VO system). The transformation function $\mathbf{g}_{jk}(\hat{\mathbf{x}}_{jk})$ is

$$\mathbf{g}_{jk}(\hat{\mathbf{x}}_{jk}) = \hat{\mathbf{T}}_k \, \hat{\mathbf{p}}_j. \tag{3.26}$$

The Gauss-Newton algorithm is based on linearized expressions that are perturbed around nominal values shown by $\bar{(\cdot)}$. Moreover, the perturbation quantities are specified by $\delta(\cdot)$ in the following computations. Define the homogeneous point operator $(\cdot)^{\boxminus}$ that transform a 4×1 vector to a 4×6 matrix by:

$$\mathbf{z}^{\boxminus} = \begin{bmatrix} \boldsymbol{\zeta} \\ \eta \end{bmatrix}^{\boxminus} := \begin{bmatrix} \eta \mathbf{1} & \boldsymbol{\zeta}^{\times} \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix}, \tag{3.27}$$

where $\boldsymbol{\zeta}$ is a 3×1 vector and η is a scalar. It can be shown that for any 6×1 vector $\mathbf{w}$,

$$\mathbf{w}^{\boxplus}\mathbf{z} \equiv -\mathbf{z}^{\boxminus}\mathbf{w}. \tag{3.28}$$

Suppose that $\bar{\mathbf{x}}_{jk}$ is perturbed to $\hat{\mathbf{x}}_{jk}$ by $\delta\hat{\mathbf{x}}_{jk}$. The nominal variables satisfy

$$\bar{\mathbf{g}}_{jk}(\bar{\mathbf{x}}_{jk}) := \bar{\mathbf{T}}_k \bar{\mathbf{p}}_j. \tag{3.29}$$

We can now perform the linearization of $\mathbf{g}_{jk}(\hat{\mathbf{x}}_{jk})$ around $\bar{\mathbf{x}}_{jk}$:

$$\begin{aligned}
\mathbf{g}_{jk}(\bar{\mathbf{x}}_{jk} + \delta\hat{\mathbf{x}}_{jk}) &\approx \left(\mathbf{1} - \delta\hat{\boldsymbol{\pi}}_k^{\boxplus}\right) \bar{\mathbf{T}}_k \left(\bar{\mathbf{p}}_j + \mathbf{P}\, \delta\hat{\boldsymbol{\epsilon}}_j\right) \\
&\approx \bar{\mathbf{T}}_k \bar{\mathbf{p}}_j - \delta\hat{\boldsymbol{\pi}}_k^{\boxplus} \bar{\mathbf{T}}_k \bar{\mathbf{p}}_j + \bar{\mathbf{T}}_k \mathbf{P}\, \delta\hat{\boldsymbol{\epsilon}}_j - \delta\hat{\boldsymbol{\pi}}_k^{\boxplus} \bar{\mathbf{T}}_k \mathbf{P}\, \delta\hat{\boldsymbol{\epsilon}}_j \\
&\approx \bar{\mathbf{T}}_k \bar{\mathbf{p}}_j - \delta\hat{\boldsymbol{\pi}}_k^{\boxplus} \bar{\mathbf{T}}_k \bar{\mathbf{p}}_j + \bar{\mathbf{T}}_k \mathbf{P}\, \delta\hat{\boldsymbol{\epsilon}}_j \\
&\approx \bar{\mathbf{T}}_k \bar{\mathbf{p}}_j + (\bar{\mathbf{T}}_k \bar{\mathbf{p}}_j)^{\boxminus} \delta\hat{\boldsymbol{\pi}}_k + \bar{\mathbf{T}}_k \mathbf{P}\, \delta\hat{\boldsymbol{\epsilon}}_j \\
&= \bar{\mathbf{g}}_{jk} + \mathbf{G}_{jk}\, \delta\hat{\mathbf{x}}_{jk},
\end{aligned} \tag{3.30}$$

where the second order terms have been dropped and

$$\mathbf{G}_{jk} \;:=\; \left[\left(\bar{\mathbf{T}}_k\bar{\mathbf{p}}_j\right)^{\boxminus} \quad \bar{\mathbf{T}}_k\mathbf{P}\right]. \tag{3.31}$$

In the next step, the camera model used to convert the landmark coordinates to the image coordinates is linearized. For this, we compose the camera model $\mathbf{f}(\cdot)$ from (2.18) with the nonlinear transformation function $\mathbf{g}_{jk}$ as follows:

$$\mathbf{h}_{jk}(\hat{\mathbf{x}}_{jk}) = (\mathbf{f}\circ\mathbf{g}_{jk})(\hat{\mathbf{x}}_{jk}) = \mathbf{f}(\mathbf{g}_{jk}(\hat{\mathbf{x}}_{jk})). \tag{3.32}$$

The nominal variables satisfy

$$\bar{\mathbf{h}}_{jk}(\bar{\mathbf{x}}_{jk}) := \mathbf{f}(\bar{\mathbf{g}}_{jk}) \tag{3.33}$$

To linearize the nonlinear function $\mathbf{h}_{jk}(\hat{\mathbf{x}}_{jk})$ around $\bar{\mathbf{x}}_{jk}$, the transformation function $\mathbf{g}_{jk}(\hat{\mathbf{x}}_{jk})$ from (3.30) is plugged into the camera model $\mathbf{f}(\cdot)$ using chain rule for the first derivatives (see (3.32)):

$$\begin{aligned}
\mathbf{h}_{jk}(\bar{\mathbf{x}}_{jk}+\delta\hat{\mathbf{x}}_{jk}) &= \mathbf{f}\big(\mathbf{g}_{jk}(\bar{\mathbf{x}}_{jk}+\delta\hat{\mathbf{x}}_{jk})\big)\\
&\approx \mathbf{f}(\bar{\mathbf{g}}_{jk}+\underbrace{\mathbf{G}_{jk}\delta\hat{\mathbf{x}}_{jk}}_{\delta\mathbf{g}_{jk}})\\
&\approx \mathbf{f}(\bar{\mathbf{g}}_{jk})+\mathbf{F}_{jk}\,\delta\mathbf{g}_{jk}\\
&\approx \bar{\mathbf{h}}_{jk}+\mathbf{H}_{jk}\,\delta\hat{\mathbf{x}}_{jk},
\end{aligned} \tag{3.34}$$

where we define the following:

$$\mathbf{H}_{jk} \;:=\; \mathbf{F}_{jk}\mathbf{G}_{jk}, \quad \mathbf{F}_{jk} := \left.\frac{\partial\mathbf{f}}{\partial\mathbf{g}}\right|_{\bar{\mathbf{g}}_{jk}} = \mathbf{M}\begin{bmatrix} \frac{1}{\bar{g}_3} & 0 & -\frac{\bar{g}_1}{(\bar{g}_3)^2} & 0\\ 0 & \frac{1}{\bar{g}_3} & -\frac{\bar{g}_2}{(\bar{g}_3)^2} & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & -\frac{\bar{g}_4}{(\bar{g}_3)^2} & \frac{1}{\bar{g}_3}\end{bmatrix}_{jk}. \tag{3.35}$$

At this step we compute the quantities associated with the previous time frame k'. The nonlinear function $\mathbf{h}_{jk'}$ is

$$\mathbf{h}_{jk'}(\hat{\mathbf{x}}_{jk'}) = \mathbf{f}(\hat{\mathbf{p}}_j), \tag{3.36}$$

and the nominal variables satisfy

$$\bar{\mathbf{h}}_{jk'} \;:=\; \mathbf{f}(\bar{\mathbf{p}}_j). \tag{3.37}$$

It is assumed that in the frame-to-frame VO the previous frame is fixed at time k', thus, no perturbations exist regarding the camera transformation. This means that only the perturbations of the location of landmarks illustrated in (3.24) are accounted in the computations. Suppose $\bar{\boldsymbol{\epsilon}}_j$ is perturbed to $\hat{\boldsymbol{\epsilon}}_j$ by $\delta\hat{\boldsymbol{\epsilon}}_j$. Now we linearize $\mathbf{h}_{jk'}(\hat{\boldsymbol{\epsilon}}_j)$ around $\bar{\boldsymbol{\epsilon}}_j$:

$$\begin{aligned}
\mathbf{h}_{jk'}(\bar{\mathbf{x}}_{jk'} + \delta\hat{\mathbf{x}}_{jk'}) &= \mathbf{f}\big(\bar{\mathbf{p}}_j + \mathbf{P}\,\delta\hat{\boldsymbol{\epsilon}}_j\big) \\
&\approx \mathbf{f}(\bar{\mathbf{p}}_j) + \mathbf{F}_{jk'}\mathbf{P}\delta\hat{\boldsymbol{\epsilon}}_j \\
&= \mathbf{f}(\bar{\mathbf{p}}_j) + \mathbf{F}_{jk'}\mathbf{G}_{jk'}\delta\hat{\mathbf{x}}_{jk'} \\
&= \bar{\mathbf{h}}_{jk'} + \mathbf{H}_{jk'}\,\delta\hat{\mathbf{x}}_{jk'},
\end{aligned} \tag{3.38}$$

and the quantities used in above computations are defined as:

$$\mathbf{H}_{jk'} \;:=\; \mathbf{F}_{jk'}\mathbf{G}_{jk'}, \quad \mathbf{F}_{jk'} := \left.\frac{\partial \mathbf{f}}{\partial \mathbf{p}}\right|_{\bar{\mathbf{p}}_j} = \mathbf{M}\begin{bmatrix} \frac{1}{\bar{p}_3} & 0 & -\frac{\bar{p}_1}{(\bar{p}_3)^2} & 0 \\ 0 & \frac{1}{\bar{p}_3} & -\frac{\bar{p}_2}{(\bar{p}_3)^2} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{\bar{p}_4}{(\bar{p}_3)^2} & \frac{1}{\bar{p}_3} \end{bmatrix}_j. \tag{3.39}$$

Thus, the nonlinear measurement function is expressed as the function $\mathbf{h}(\cdot)$ and the

Jacobian matrix including the first derivatives is referred to as $\mathbf{H}(\cdot)$.

After linearizing $\mathbf{h}_{jk'}$ and $\mathbf{h}_{jk}$, the linear approximations of the error terms described in (3.19) and (3.20) are given by

$$\mathbf{e}_{jk}(\hat{\boldsymbol{x}}_{jk}) \approx \mathbf{y}_{jk} - \bar{\mathbf{h}}_{jk} - \mathbf{H}_{jk}\,\delta\hat{\boldsymbol{x}}_{jk} := \bar{\mathbf{e}}_{jk} - \mathbf{H}_{jk}\,\delta\hat{\boldsymbol{x}}_{jk}, \tag{3.40}$$

$$\mathbf{e}_{jk'}(\hat{\boldsymbol{x}}_{jk'}) \approx \mathbf{y}_{jk'} - \bar{\mathbf{h}}_{jk'} - \mathbf{H}_{jk'}\,\delta\hat{\boldsymbol{x}}_{jk'} := \bar{\mathbf{e}}_{jk'} - \mathbf{H}_{jk'}\,\delta\hat{\boldsymbol{x}}_{jk'}. \tag{3.41}$$

The nominal full state $\bar{\boldsymbol{x}}_k$ is perturbed to $\hat{\boldsymbol{x}}_k$ by the overall state perturbation $\delta\hat{\boldsymbol{x}}_k$ given by

$$\delta\hat{\boldsymbol{x}}_k = \begin{bmatrix} \delta\hat{\boldsymbol{\pi}}_k \\ \delta\hat{\boldsymbol{\epsilon}}_1 \\ \vdots \\ \delta\hat{\boldsymbol{\epsilon}}_J \end{bmatrix}. \tag{3.42}$$

To write the objective function Q_{ba} (see (3.21)) in terms of $\delta\hat{\boldsymbol{x}}_k$, we need to use projection matrices $\mathbf{P}_{jk}$ and $\mathbf{P}_{jk'}$ that obtain $\delta\hat{\boldsymbol{x}}_{jk}$ and $\delta\hat{\boldsymbol{x}}_{jk'}$ components from $\delta\hat{\boldsymbol{x}}_k$. The projection matrices are obtained by

$$\delta\hat{\boldsymbol{x}}_{jk} := \mathbf{P}_{jk}\,\delta\hat{\boldsymbol{x}}_k, \qquad \delta\hat{\boldsymbol{x}}_{jk'} := \mathbf{P}_{jk'}\,\delta\hat{\boldsymbol{x}}_k, \tag{3.43}$$

where $\delta\hat{\boldsymbol{x}}_{jk}$ and $\delta\hat{\boldsymbol{\epsilon}}_j$ are described in (3.22). Define the following quantities

$$\bar{Q}_{\text{ba}} := \frac{1}{2}\sum_j \left(\bar{\mathbf{e}}_{jk}^T \mathbf{R}_{jk}^{-1} \bar{\mathbf{e}}_{jk} + \bar{\mathbf{e}}_{jk'}^T \mathbf{R}_{jk'}^{-1} \bar{\mathbf{e}}_{jk'} \right), \tag{3.44}$$

$$\mathbf{q}_{\text{ba}} := -\sum_j \left(\mathbf{P}_{jk}^T \mathbf{H}_{jk}^T \mathbf{R}_{jk}^{-1} \bar{\mathbf{e}}_{jk} + \mathbf{P}_{jk'}^T \mathbf{H}_{jk'}^T \mathbf{R}_{jk'}^{-1} \bar{\mathbf{e}}_{jk'} \right), \tag{3.45}$$

$$\mathbf{Q}_{\text{ba}} := \sum_j \left(\mathbf{P}_{jk}^T \mathbf{H}_{jk}^T \mathbf{R}_{jk}^{-1} \mathbf{H}_{jk} \mathbf{P}_{jk} + \mathbf{P}_{jk'}^T \mathbf{H}_{jk'}^T \mathbf{R}_{jk'}^{-1} \mathbf{H}_{jk'} \mathbf{P}_{jk'} \right). \tag{3.46}$$

The linear approximations of error terms given by (3.40) and (3.41) are substituted

into the expression of the objective function Q_{ba}. The final expression is written in terms of $\delta\hat{\boldsymbol{x}}_k$ using (3.42) to (3.46) as follows:

$$\begin{aligned} Q_{\text{ba}}(\bar{\boldsymbol{x}}_k + \delta\hat{\boldsymbol{x}}_k) &:= \frac{1}{2}\sum_j \Big(\mathbf{e}_{jk}(\hat{\boldsymbol{x}}_{jk})^T \mathbf{R}_{jk}^{-1} \mathbf{e}_{jk}(\hat{\boldsymbol{x}}_{jk}) + \mathbf{e}_{jk'}(\hat{\boldsymbol{x}}_{jk'})^T \mathbf{R}_{jk'}^{-1} \mathbf{e}_{jk'}(\hat{\boldsymbol{x}}_{jk'}) \Big) \\ &\approx \frac{1}{2}\sum_j \Big((\bar{\mathbf{e}}_{jk} - \mathbf{H}_{jk}\,\mathbf{P}_{jk}\,\delta\hat{\boldsymbol{x}}_k)^T \mathbf{R}_{jk}^{-1} (\bar{\mathbf{e}}_{jk} - \mathbf{H}_{jk}\,\mathbf{P}_{jk}\,\delta\hat{\boldsymbol{x}}_k) + \\ &\quad (\bar{\mathbf{e}}_{jk'} - \mathbf{H}_{jk'}\,\mathbf{P}_{jk'}\,\delta\hat{\boldsymbol{x}}_k)^T \mathbf{R}_{jk'}^{-1} (\bar{\mathbf{e}}_{jk'} - \mathbf{H}_{jk'}\,\mathbf{P}_{jk'}\,\delta\hat{\boldsymbol{x}}_k) \Big) \\ &\approx \bar{Q}_{\text{ba}} + \mathbf{q}_{\text{ba}}^T\,\delta\hat{\boldsymbol{x}}_k + \frac{1}{2}\delta\hat{\boldsymbol{x}}_k^T\,\mathbf{Q}_{\text{ba}}\,\delta\hat{\boldsymbol{x}}_k. \end{aligned} \tag{3.47}$$

After elaborating on the steps to construct the objective function $Q_{\text{ba}}(\bar{\boldsymbol{x}}_k + \delta\hat{\boldsymbol{x}}_k)$, we now take its derivative using (3.47) in order to minimize the function with respect to $\delta\hat{\boldsymbol{x}}_k$:

$$\frac{\partial^T Q_{\text{ba}}(\bar{\boldsymbol{x}}_k + \delta\hat{\boldsymbol{x}}_k)}{\partial\,\delta\hat{\boldsymbol{x}}_k} = \mathbf{q}_{\text{ba}} + \mathbf{Q}_{\text{ba}}\,\delta\hat{\boldsymbol{x}}_k. \tag{3.48}$$

We continue by setting (3.48) to zero and finding the solution of the following linear system that is the optimal perturbation, $\delta\hat{\boldsymbol{x}}_k^\star$, of the system:

$$-\mathbf{Q}_{\text{ba}}\delta\hat{\boldsymbol{x}}_k^\star = \mathbf{q}_{\text{ba}}. \tag{3.49}$$

3.2.4 Iteration Scheme

The Gauss-Newton algorithm is an iterative algorithm for which we define the iteration number by the notation (i). An iterative algorithm starts by a set of initial guesses and iterates using some update equations until convergence.

Initial Guess

In the first iteration, the algorithm is fed by some initial guess for the nominal state $\{\bar{\mathbf{T}}_k, \bar{\mathbf{p}}_j\}(1)$. The initial guess for the nominal camera pose at time k is selected the

same as the camera frame at time k'

$$\bar{\mathbf{T}}_k(1) \;=\; \mathbf{1}_{4\times 4}, \tag{3.50}$$

where $\mathbf{1}_{4\times 4}$ is a 4×4 identity matrix. We know that $\mathbf{f}(\hat{\mathbf{p}}_j)$ gives a set of noise-free measurements corresponding to landmark j captured by the camera at time k'. To initiate the position of landmark j in the first iteration, the observation vector $\mathbf{y}_{jk',\text{meas}}$ is plugged into $\mathbf{f}^{-1}(\cdot)$ as follows:

$$\bar{\mathbf{p}}_j(1) \;=\; \mathbf{f}^{-1}(\mathbf{y}_{jk'\text{meas}}). \tag{3.51}$$

Iteration

In the i-th iteration, the optimal perturbation $\delta\hat{\boldsymbol{x}}_k^\star(i)$ is obtained by

$$-\mathbf{Q}_{\text{ba}}(i)\delta\hat{\boldsymbol{x}}_k^\star(i) = \mathbf{q}_{\text{ba}}(i), \tag{3.52}$$

where $\mathbf{Q}_{\text{ba}}(i)$ and $\mathbf{q}_{\text{ba}}(i)$ are obtained by (3.46) and (3.45) evaluated at $\bar{\boldsymbol{x}}_k(i)$ and using the measurement vectors $\mathbf{y}_{jk',\text{meas}}$ and $\mathbf{y}_{jk,\text{meas}}$. Note that the measurement vectors remain the same in all iterations. In iteration i, the optimal transformation perturbation, $\delta\hat{\mathbf{T}}_k^\star(i)$, and the optimal perturbation of landmark position, $\delta\hat{\mathbf{p}}_j^\star(i)$, which are obtained form the optimal perturbation of the full state, $\delta\hat{\boldsymbol{x}}_k^\star(i)$, are given by

$$\begin{aligned}
\delta\hat{\mathbf{T}}_k^\star(i) \;&:=\; \begin{bmatrix} \mathbf{C}_k(\delta\hat{\boldsymbol{\phi}}_k^\star) & \mathbf{S}_k(\delta\hat{\boldsymbol{\phi}}_k^\star)\,\delta\hat{\boldsymbol{\rho}}_k^\star \\ \mathbf{0}^T & 1 \end{bmatrix}(i), \\
\delta\hat{\mathbf{p}}_j^\star(i) \;&:=\; \mathbf{P}\delta\hat{\boldsymbol{\epsilon}}_j^\star(i), \quad \forall j = 1\ldots J,
\end{aligned} \tag{3.53}$$

where the optimal perturbations $\delta\hat{\boldsymbol{\rho}}_k^\star(i)$, $\delta\hat{\boldsymbol{\phi}}_k^\star(i)$, and $\delta\hat{\boldsymbol{\epsilon}}_j^\star(i)$ are extracted from $\delta\hat{\boldsymbol{x}}^\star(i)$. The matrices $\mathbf{C}_k$ and $\mathbf{S}_k$ were introduced in Chapter 2 in the expressions (2.6) and (2.7) and $\mathbf{P}$ is a projection matrix presented previously in (3.25). For more information

on the procedure of creating $\{\delta\hat{\mathbf{T}}_k, \delta\hat{\mathbf{p}}_j\}$ using perturbation in translation and rotation of the camera, $\delta\hat{\boldsymbol{\rho}}$ and $\delta\hat{\boldsymbol{\phi}}_k$, and the perturbation in the landmark position, $\delta\hat{\boldsymbol{\epsilon}}_j$, refer to Barfoot et al. [5].

The optimal perturbations of iteration i, $\{\delta\hat{\mathbf{T}}_k^\star, \delta\hat{\mathbf{p}}_j^\star\}(i)$, are used to update the nominal state of iteration i, $\{\bar{\mathbf{T}}_k, \bar{\mathbf{p}}_j\}(i)$, and produce the nominal state of the next iteration as follows:

$$\begin{aligned} \bar{\mathbf{T}}_k(i+1) &= \hat{\mathbf{T}}_k(i) \\ &= \delta\hat{\mathbf{T}}_k^\star(i)\bar{\mathbf{T}}_k(i), \end{aligned} \tag{3.54}$$

$$\begin{aligned} \bar{\mathbf{p}}_j(i+1) &= \hat{\mathbf{p}}_j(i) \\ &= \delta\hat{\mathbf{p}}_j^\star(i) + \bar{\mathbf{p}}_j(i). \end{aligned} \tag{3.55}$$

$\bar{\mathbf{T}}_k(i+1)$ and $\bar{\mathbf{p}}_j(i+1)$ are used as the start point of the next iteration.

Convergence

The algorithm is iterated by equations (3.54) and (3.55) until the following stopping criterion is met

$$|(\delta\hat{\boldsymbol{x}}_k^\star(i) - \delta\hat{\boldsymbol{x}}_k^\star(i-1))^T(\delta\hat{\boldsymbol{x}}_k^\star(i) - \delta\hat{\boldsymbol{x}}_k^\star(i-1))| < 10^{-6}. \tag{3.56}$$

The increase or decrease of the threshold value could impact the performance of the estimation algorithm. We used 10^{-6} as the threshold value in our algorithm. Once the stopping criterion is met, the last updated state is our desired state estimate, $\{\hat{\mathbf{T}}_k, \hat{\mathbf{p}}_j\}$.

3.2.5 Summary

This subsection presents our nonlinear estimator, which is based on the Gauss-Newton method, as a function. This function obtains measurements, $\mathbf{y}_{\text{meas}}$, as input and pro-

duces the state estimate, $\hat{\boldsymbol{x}}_k$, as output. In the VO estimation problem, we are only interested in the camera pose estimates. Therefore, we only indicate the transformation matrix $\hat{\mathbf{T}}_k$ or equivalently the camera pose vector $\hat{\boldsymbol{\pi}}_k$ as the output of the VO system. The nonlinear estimator is denoted by $\mathbf{h}_{\text{vo}}$

$$\hat{\boldsymbol{\pi}}_k = \mathbf{h}_{\text{vo}}(\mathbf{y}_{\text{meas}}), \tag{3.57}$$

where all the landmarks that are observed by the stereo camera at time k' and time k are combined into a measurement vector as follows:

$$\mathbf{y}_{\text{meas}} := \left[\begin{array}{c} \mathbf{y}_{1k'} \\ \vdots \\ \mathbf{y}_{Jk'} \\ \hline \mathbf{y}_{1k} \\ \vdots \\ \mathbf{y}_{Jk} \end{array}\right]_{\text{meas}} \tag{3.58}$$

Now that the algorithm of our estimator is described completely, we will state, in the next chapters, the issue regarding the presence of bias in the stereo VO problems and propose a novel technique to remove bias from the path estimates of a rover.

Chapter 4

Bias Estimation Machinery

This chapter begins with a general definition of bias in statistics and explains the bias terminology in the robotics field, specially in visual odometry problems. Section 4.2 demonstrates bias using two simplified symmetric and asymmetric cases. This is the first step in identifying bias for the VO problems in this thesis. Bias estimation machinery is described in Section 4.3. Different methods of estimating bias including the Monte Carlo method, the sigma-point technique, and the Box method are introduced in this section. In Section 4.4, the generation of truncated Gaussian statistics for use in the bias estimation algorithm is described, which accounts for rejected observations based on the disparity threshold. Bias estimation for stereo-triangulation associated with the position estimation of a single landmark is covered in Section 4.5.

4.1 Bias Terminology

In the statistics literature, an unknown parameter θ_{t} associated with a probability distribution can be estimated based on observations arising from the probability distribution, and modelled as a random vector $\boldsymbol{\gamma}$. This is performed by an estimator

$\mathbf{h}_{\text{est}}(\boldsymbol{\gamma})$. In the case that the expected value of the estimator equals the true value of the parameter θ_{t} we have unbiasedness, otherwise there is a systematic error or bias b that is defined as follows:

$$b = E[\mathbf{h}_{\text{est}}(\boldsymbol{\gamma})] - \theta_{\text{t}}, \tag{4.1}$$

where the expectation is performed with respect to the probability distribution parameterized by θ_{t} [64]. In the visual odometry problems, the motion of a mobile robot is estimated using a nonlinear estimation algorithm (Bundle Adjustment). The expected value of the state estimates may deviate from ground truth due to the nonlinearities in the model. This deviation from ground truth is called *bias* and is present for specific configurations of landmarks and the camera pose in the VO setup.

4.2 Bias Identification using Symmetric and Asymmetric Cases

To systematically identify bias in VO path estimates, we designed two simplified setups with different configurations of landmarks. In both setups the camera moves forward in the direction of $z-$axis (along horizontal) towards the landmarks. In two consecutive time frames, five landmarks are observed from two different viewpoints at time k' and time k. As the robot moves, the same configuration of landmarks with respect to the camera poses is preserved for each pair of consecutive frames. The $z-$axis of the camera frame is aligned with the traversed path and the $y-$axis points to the ground where the origin of the camera frame is located above the ground. As already pointed out above, we consider two setups to demonstrate that bias appears in the estimates when we change the configuration of landmarks. The

first setup describes a symmetric configuration in which four landmarks are located at corners of a hypothetical rectangle on a plane perpendicular to $z-$axis positioned at some positive z (symmetric with respect to $x-$ and $y-$ axes), and one landmark is located on the $z-$axis closer to the camera frame than the plane of four landmarks (to facilitate bundle adjustment). This configuration of landmarks with respect to the camera frames provides a symmetric pattern in image space; thus, it is referred to as the symmetric configuration. In the asymmetric configuration (in image space) the plane of landmarks is tilted in a way that the pair of top landmarks become farther than the bottom ones with respect to the camera frame. The asymmetric case is representative of many real-world situations; for example, a planetary rover traversing a surface typically has far-away landmarks in the top of its images and close ones in the bottom. The symmetric and asymmetric configurations are shown in Figures 4.1 and 4.2. The $x-$, $y-$, and $z-$ axes of the camera frame are shown in green, blue, and red in the figures (the frame at previous time is faded).

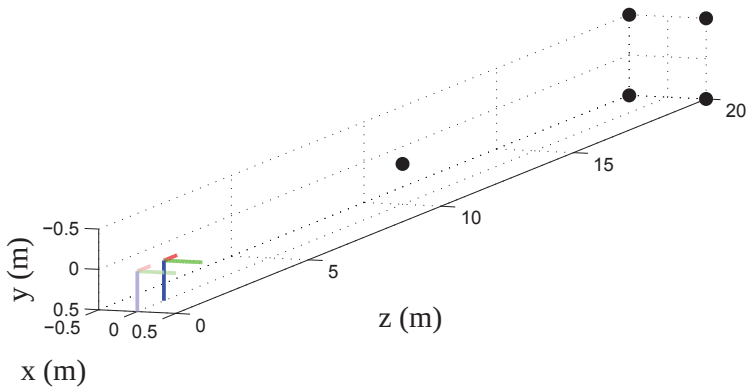

Figure 4.1: A symmetric configuration of landmarks in 3D space along with the camera frame at the previous time and current time (camera moves one metre in the $z-$axis direction.)

It is verified by Monte Carlo simulations that the asymmetric configuration re-

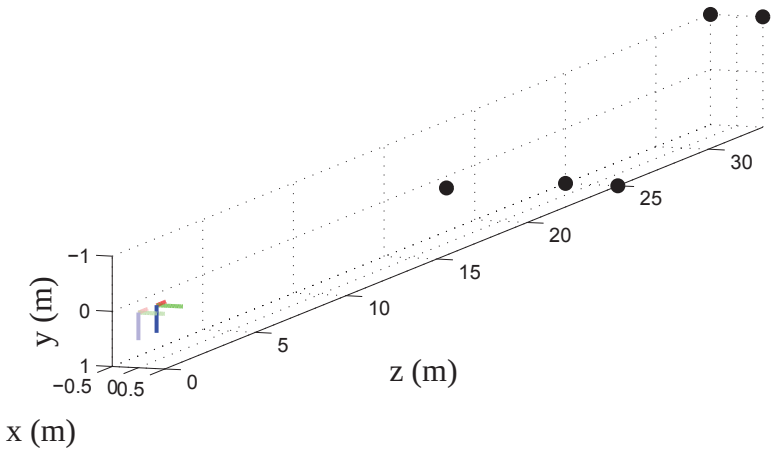

Figure 4.2: An asymmetric configuration of landmarks in 3D space along with the camera frame at the previous time and current time (camera moves one metre in the $z-$axis direction.)

sults in biased camera path estimates, whereas the symmetric configuration produces unbiased estimates. The most frequent types of image noise can be represented by the Gaussian distribution. Therefore, to compare the results obtained for symmetric and asymmetric cases, we generated a large number of different noisy measurement samples by adding Gaussian noise to the noise-free landmark observations (in image space). Then, we passed these noisy measurement samples through the nonlinear estimator in the VO problem, $\mathbf{h}_{\text{vo}}(\cdot)$ (see Subsection 3.2.5), to produce camera path estimates. Note that for illustration we have compounded the 1-metre results (obtained for one pair of time frames) 500 times to show what the effect of the bias would be over a realistic distance. Figure 4.3 shows the mean of these path estimates. The simulation result shows that there is a significant bias in the direction of $y-$axis in the asymmetric case, which grows in a superlinear manner as the camera travels. Hence, certain configurations of landmarks could give rise to bias in estimates. Monte Carlo simulation is computationally extremely expensive, therefore we seek a more efficient method of estimating bias. The algorithms of estimating bias using the Monte Carlo

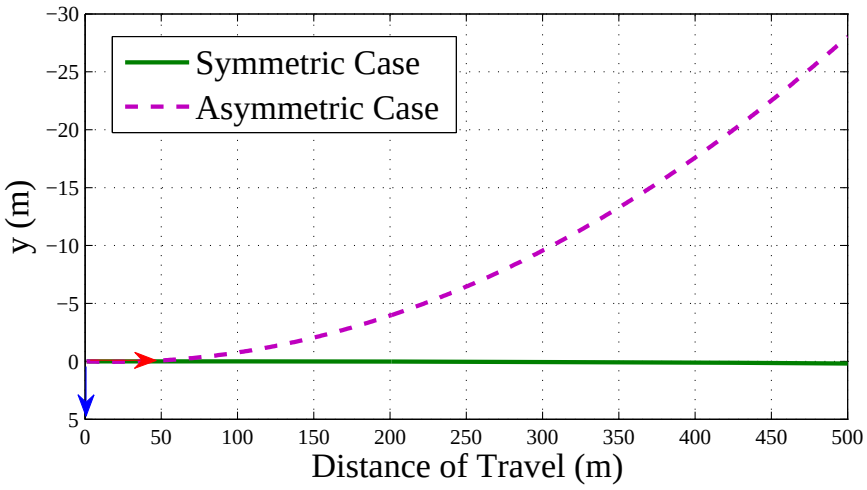

Figure 4.3: Path estimates for symmetric and asymmetric cases based on brute-force Monte Carlo simulations for the purpose of bias identification.

method and other methods are described in more detail in Section 4.3.

4.3 Bias Estimation Machinery

In this section, we describe an approach to quantify bias in a generic nonlinear estimation problem. To this end, we consider the estimator as a black box and analyse its output when we vary the noise on its input measurements.

We assume that we have an estimator, $\mathbf{h}_{\text{est}}(\cdot)$, that takes measurements, $\mathbf{y}_{\text{meas}}$, with covariance matrix, $\mathbf{R}$, as input and returns a state estimate, $\hat{\mathbf{x}}$, as output:

$$\hat{\mathbf{x}} = \mathbf{h}_{\text{est}}(\mathbf{y}_{\text{meas}}). \tag{4.2}$$

The following expression describes bias assuming that we know the true state, $\mathbf{x}_{\text{t}}$:

$$\mathbf{b} = E[\hat{\mathbf{x}}] - \mathbf{x}_{\text{t}}. \tag{4.3}$$

To start the process of quantifying bias we generate noise-free measurements, $\mathbf{y}_{\mathrm{t}}$, using our nonlinear measurement function (e.g., our camera model from (2.18)), $\mathbf{f}(\cdot)$, and the state, $\mathbf{x}_{\mathrm{t}}$:

$$\mathbf{y}_{\mathrm{t}} = \mathbf{f}(\mathbf{x}_{\mathrm{t}}). \tag{4.4}$$

A brute-force method of estimating the bias is to use Monte Carlo simulation where we vary the noise on the measurements; to do so, we draw a large number, N, of random samples from zero-mean Gaussian distribution,

$$\mathbf{y}_i \leftarrow \mathbf{y}_{\mathrm{t}} + \mathbf{n}_i, \qquad \mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad 1 \leq i \leq N. \tag{4.5}$$

We then run the estimator for each measurement sample and average the results to compute the bias. Clearly, the Monte Carlo approach will be computationally expensive since a large number of samples is needed. For instance, the number of samples for the simulation shown in Figure 4.3 was $N = 20000$. Accordingly, we propose a second sampling approach based on the sigma-point method [54]. The so-called sigma-point approach is a proven method that is widely used in nonlinear filtering (such as sigma-point Kalman filters) [53, 55, 63, 72, 75, 76]. As an example, the sigma-point method is utilized by Coogle et al. [21] to reduce bias during measurement conversion from sine-space coordinates to Cartesian coordinates in a long-range radar problem. The sigma-point method is computationally efficient since it utilizes a discrete sampling distribution with the same first and second moments as the original distribution. The samples from this discrete distribution are called sigma-points and will be transformed by the nonlinear estimator. The sigma-points are chosen to lie on the principal component of the covariance matrix, $\mathbf{R}$, in addition to one point

that represents the mean of the distribution [89]. To do so, we first compute $\mathbf{S}_{\text{ch}}$, the Cholesky factor of the measurement noise covariance matrix, $\mathbf{R}$:

$$\mathbf{S}_{\text{ch}}\,\mathbf{S}_{\text{ch}}^{T} = \mathbf{R}, \qquad \mathbf{S}_{\text{ch}} = [\mathbf{s}_1 \cdots \mathbf{s}_L]. \tag{4.6}$$

Moreover, the minimal number of sigma-points that maintain information associated with the sampling distribution is $M = 2L+1$ where L is the length of the observation vector $\mathbf{y}_{\text{t}}$. Prior to determining the sigma-points, we define the weighting factors ω_i:

$$\omega_i = \begin{cases} \frac{\lambda}{L+\lambda} & \text{if } i = 1 \\ \frac{1}{[2(L+\lambda)]} & \text{otherwise} \end{cases} \qquad i = 1 \ldots M, \tag{4.7}$$

where $\lambda = \alpha^2(L+\kappa) - L$, κ is normally 0, and $\alpha \in (0.1, 1)$ is a sigma-point *spreading parameter* [69]. The set of sigma-points, which is symmetric, is given by:

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{y}_{\text{t}} \\ \mathbf{y}_{l+1} &= \mathbf{y}_{\text{t}} + \tfrac{1}{\sqrt{2\omega_{l+1}}}\mathbf{s}_l \qquad\qquad l = 1 \ldots L. \\ \mathbf{y}_{l+L+1} &= \mathbf{y}_{\text{t}} - \tfrac{1}{\sqrt{2\omega_{l+L+1}}}\mathbf{s}_l \end{aligned} \tag{4.8}$$

Regardless of the sampling approach, we then pass all the samples through the non-linear estimator,

$$\hat{\mathbf{x}}_i = \mathbf{h}_{\text{est}}(\mathbf{y}_i), \tag{4.9}$$

and combine them to compute the bias as follows:

$$\mathbf{b} \;\approx\; \sum_{i=1}^{I} \omega_i\, \hat{\mathbf{x}}_i - \mathbf{x}_{\text{t}}, \tag{4.10}$$

where $I = N$ and $\omega_i = \frac{1}{N}$ in the Monte Carlo approach and $I = M$ and ω_i are given

in (4.7) for the sigma-point approach.

Box [14] provides a third method that is analytical rather than sample-based. In the method proposed by Box, the expected bias in estimates achieved by maximum likelihood is calculated quantitatively using Taylor series expansion up to second order [14]. In the pose optimization step of VO, we seek to minimize a negative-log-likelihood criterion:

$$Q = \frac{1}{2}(\mathbf{y} - \mathbf{f}(\hat{\mathbf{x}}))^T \mathbf{R}^{-1}(\mathbf{y} - \mathbf{f}(\hat{\mathbf{x}})), \tag{4.11}$$

where $\hat{\mathbf{x}}$ is the state estimate and $\mathbf{y}$ satisfies the measurement model (4.5). Define $\mathbf{W}$ by

$$\mathbf{W} := \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}, \tag{4.12}$$

where $\mathbf{H}$ is the Jacobian of $\mathbf{f}(\cdot)$ (evaluated at $\mathbf{x}_\mathrm{t}$). Then Box shows that the bias is given by

$$E[\hat{\mathbf{x}}] - \mathbf{x}_\mathrm{t} \approx -\frac{1}{2}\mathbf{W}^{-1}\mathbf{H}^T\mathbf{R}^{-1}\sum_{i=1}^{P} \mathbf{1}_i \operatorname{tr}(\boldsymbol{\mathcal{H}}_i \mathbf{W}^{-1}), \tag{4.13}$$

where P is the dimension of our state vector, $\mathbf{1}_i$ is the ith column of the identity matrix, and $\boldsymbol{\mathcal{H}}_i$ is the Hessian of the ith row of $\mathbf{f}(\cdot)$ (evaluated at $\mathbf{x}_\mathrm{t}$) [14]. Further details on how to use (4.13) is covered in Section 5.1.1 for VO problems.

4.4 Truncated Gaussian Statistics for Bias Estimation

A disparity threshold is used in our setup to assure the validity of the stereo observations. A truncation is applied based on the disparity threshold in a way that

measurements with too small or negative disparity threshold (corresponding to the landmarks located too far from the camera or behind it) are discarded from the set of noisy measurements that are produced based on a Gaussian distribution. This has the effect of modifying the original noise distribution. Therefore, we attempt to refit a new Gaussian to the measurement samples from Monte Carlo simulations that are filtered by the disparity threshold (i.e. samples that are valid). This is clarified through an example of imposing truncation on a large number of samples using a disparity threshold of 4 pixels and the result is depicted in Figure 4.4. It is observed that during this process, many samples are truncated and so they will not be appropriate for use in our estimation algorithm. We also note that, in this example, the original noise properties were uncorrelated between the left/right images, but after applying the disparity threshold they are now correlated.

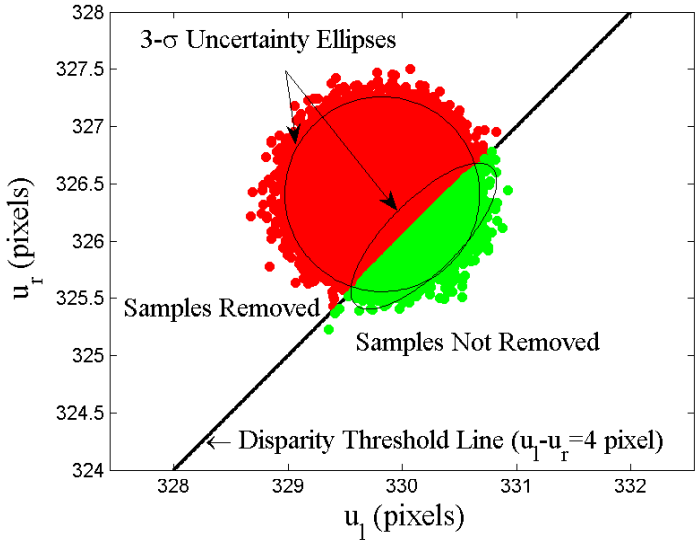

Figure 4.4: A disparity threshold is used to ensure stereo observations are valid (i.e., not too far away and not behind the camera). With Gaussian noise added to the noise-free landmark observations, it is viewed that many of the noisy measurements (10000 sample used) fall on the 'bad' (red) side of the disparity threshold; this has the effect of altering the effective noise properties of the camera. In the bias estimation algorithm, we refit a new Gaussian based on the samples that pass (green) the disparity threshold.

We now propose two different flavours of the sigma-point and Box methods that take into account the effect of sample truncation imposed by the disparity threshold. As shown in Figure 4.4, a new Gaussian is refitted to the valid observations. Henceforth, we use subscript 'trun' for the quantities that are computed using the set of measurements that are filtered with the truncation operation. In order to use the truncated Gaussian in the estimation algorithm we compute its mean, $\mathbf{y}^{\text{mc}}_{\text{m,trun}}$, and covariance, $\mathbf{R}^{\text{mc}}_{\text{trun}}$, by

$$\mathbf{y}^{\text{mc}}_{\text{m,trun}} = \frac{1}{N_{\text{trun}}} \sum_{i=1}^{N_{\text{trun}}} \mathbf{y}^{\text{mc}}_{i,\text{trun}}, \tag{4.14}$$

$$\mathbf{R}^{\text{mc}}_{\text{trun}} = \frac{1}{N_{\text{trun}}} \sum_{i=1}^{N_{\text{trun}}} (\Delta\mathbf{y}^{\text{mc}}_{i,\text{trun}}) (\Delta\mathbf{y}^{\text{mc}}_{i,\text{trun}})^T, \tag{4.15}$$

where N_{trun} is the number of Monte Carlo samples remaining after truncation and $\Delta\mathbf{y}^{\text{mc}}_{i,\text{trun}} = \mathbf{y}^{\text{mc}}_{i,\text{trun}} - \mathbf{y}^{\text{mc}}_{\text{m,trun}}$.

The modified sigma-point method which is referred to as the *truncated sigma-point* method replaces $\mathbf{R}$ by $\mathbf{R}^{\text{mc}}_{\text{trun}}$ in (4.6) and substitutes $\mathbf{y}_{\text{t}}$ by $\mathbf{y}^{\text{mc}}_{\text{m,trun}}$ in (4.8). The parameter λ in the sigma-point approach is tuned to guarantee that all the generated sigma-points pass the disparity threshold test. Then, we repeat all the steps described in Section 4.3 to estimate the bias. The modified version of Box method which is called the *truncated Box* method replaces the covariance matrix, $\mathbf{R}$, by $\mathbf{R}^{\text{mc}}_{\text{trun}}$ and evaluates the Jacobian and Hessian matrices at $\hat{\mathbf{x}}^{\text{mc}}_{\text{m,trun}} = \mathbf{f}^{-1}(\mathbf{y}^{\text{mc}}_{\text{m,trun}})$ instead of the true state, $\mathbf{x}_{\text{t}}$, where $\mathbf{f}^{-1}$ is the inverse the of camera model described in (2.20). In the results, we will denote these modified methods with subscript 'trun' to indicate that they use the truncated Gaussian statistics to account for the disparity threshold. To be clear, although we make use of the samples generated by Monte Carlo to produce a Gaussian distribution that reflects the effect of the disparity threshold, the Monte Carlo samples

are not passed through the nonlinear estimator in the modified sigma-point or Box methods; they are only used to generate modified measurement statistics.

4.5 Bias Estimation for Stereo-Triangulation

Before developing the bias estimation algorithm for VO problems, we first discuss how to estimate bias in a simpler problem. In this section, we investigate the basic problem of estimating bias in the position of one landmark using one stereo observation. In this problem, the stereo camera is stationary; therefore, (2.16) simplifies to $\mathbf{g}_j = \mathbf{p}_j$. In addition, we assume that we know the true position of the landmark,

$$\mathbf{p}_{\mathrm{t}} = \begin{bmatrix} \boldsymbol{\epsilon}_{\mathrm{t}} \\ 1 \end{bmatrix}, \tag{4.16}$$

where $\boldsymbol{\epsilon}_{\mathrm{t}} = (x_{\mathrm{t}}, y_{\mathrm{t}}, z_{\mathrm{t}})$ presents the true position of the landmark in 3D space. The noise-free observation of the landmark using camera model $\mathbf{f}(\cdot)$ (refer to (2.17)) is given by

$$\mathbf{y}_{\mathrm{t}} = \mathbf{f}(\mathbf{p}_{\mathrm{t}}). \tag{4.17}$$

We next provide details on how each of the three bias estimation techniques discussed in Section 4.3 can be used in this scenario.

To estimate bias using the Monte Carlo method, we use (4.5) to generate a set of noisy measurements, $\mathbf{y}_i$, $i = 1 \dots M$. Also, the covariance matrix, $\mathbf{R}$, is obtained using (3.17). For simplicity, we assume $\sigma_{u_l} = \sigma_{v_l} = \sigma_{u_r} = \sigma_{v_r} = \sigma$, where σ is the standard deviation of noise added to each component of the vector $\mathbf{y}_{\mathrm{t}}$ in image space.

As presented in Figure 4.4, we then truncate (i.e., discard) the measurements

with disparities less than the disparity threshold. Next, we pass each non-truncated measurement, $\mathbf{y}_i$, through the inverse of the camera model, $\mathbf{f}^{-1}(\cdot)$ from (2.20), to obtain its corresponding 3×1 landmark position, $\boldsymbol{\epsilon}_i$, as follows:

$$\hat{\mathbf{p}}_i = \begin{bmatrix} \hat{\boldsymbol{\epsilon}}_i \\ 1 \end{bmatrix} = \mathbf{f}^{-1}(\mathbf{y}_i). \tag{4.18}$$

Using (4.10), the bias in the position of the landmark based on Monte Carlo simulations is given by

$$\begin{aligned} \mathbf{b} &= E[\hat{\boldsymbol{\epsilon}}] - \boldsymbol{\epsilon}_{\mathrm{t}} \\ &\approx \sum_{i=1}^{I} \omega_i \, \hat{\boldsymbol{\epsilon}}_i - \boldsymbol{\epsilon}_{\mathrm{t}}, \end{aligned} \tag{4.19}$$

where $I = N_{\mathrm{trun}}$ (see Subsection 4.4) and $\omega_i = \frac{1}{N_{\mathrm{trun}}}$ for the Monte Carlo method.

For the sigma-point method, we use sigma-points, $\mathbf{y}_i$, given in (4.8). If the truncated sigma-point method is used for bias estimation, the truncated Gaussian statistics are utilized to generate sigma-points as described in Subsection 4.4. Afterwards, we pass each sigma-point, which is ensured to have a disparity larger than the disparity threshold by tuning λ, through $\mathbf{f}^{-1}(\cdot)$ to obtain its corresponding 3×1 landmark position, $\hat{\boldsymbol{\epsilon}}_i$. The bias is then computed from (4.19), where $I = M$, the number of sigma-points, and the ω_i are calculated using (4.7).

For the method of Box, define the Jacobian and the Hessian evaluated at $\mathbf{p}_{\mathrm{t}}$ denoted by $\mathbf{F}$ and $\boldsymbol{\mathcal{F}}$:

$$\mathbf{F} := \left.\frac{\partial \mathbf{f}}{\partial \mathbf{p}}\right|_{\mathbf{p}_{\mathrm{t}}}, \qquad \boldsymbol{\mathcal{F}} := \left.\frac{\partial^2 \mathbf{f}_i}{\partial \mathbf{p} \partial \mathbf{p}^T}\right|_{\mathbf{p}_{\mathrm{t}}}, \tag{4.20}$$

where i is an index over the rows of $\mathbf{f}(\cdot)$. The projection matrix $\mathbf{P}$, which was used

in Chapter 3 to render the landmark position in a 3×1 vector form, $\boldsymbol{\epsilon} = (x, y, z)$, defined in (3.25) is utilized to produce the Jacobian and the Hessian evaluated at $\boldsymbol{\epsilon}_{\mathrm{t}}$

$$\mathbf{H} := \mathbf{F}\,\mathbf{P}, \qquad \boldsymbol{\mathcal{H}} := \mathbf{P}^T \boldsymbol{\mathcal{F}}\,\mathbf{P}, \tag{4.21}$$

Next, $\mathbf{W}$ is computed using (4.12) and finally all the above matrices along with the covariance matrix, $\mathbf{R}$, from (4.5) are substituted into (4.13) and the bias, $\mathbf{b} = E[\hat{\boldsymbol{\epsilon}}] - \boldsymbol{\epsilon}_{\mathrm{t}}$, is calculated. If the truncated Box method is used, the matrices in (4.20) are all evaluated at $\hat{\mathbf{p}}^{\mathrm{mc}}_{\mathrm{m,trun}} = \mathbf{f}^{-1}(\mathbf{y}^{\mathrm{mc}}_{\mathrm{m,trun}})$ and $\mathbf{R}^{\mathrm{mc}}_{\mathrm{trun}}$ is used for bias estimation.

4.5.1 Simulation Results for Bias Estimation

In this section, to see the effect of increasing the nonlinearity of the problem, we consider one landmark at the location $(1, -5, z)$ in the fixed camera frame where z varies (see Figure 4.5 for the structure of the setup).

Figure 4.6 illustrates the bias estimated by five different methods: the Monte Carlo method, the sigma-point method (two versions), and the Box method (two versions). It is interesting to observe that all the methods are quite similar until the disparity threshold starts to remove a large number of samples and then only the truncated sigma-point method (with the truncated Gaussian statistics) performs as well as Monte Carlo (considered to be our ground truth). Based on the bottom plot in Figure 4.6, the difference between the bias estimated by the truncated sigma-point and Monte-Carlo is only 3.7×10^{-4} metres when $z = 22$ metres. Although the truncated Box method performs better than the Box method, it cannot track ground truth the same way that the truncated sigma-point does (when the same samples become truncated using the disparity threshold). It is worth noting that the estimated bias

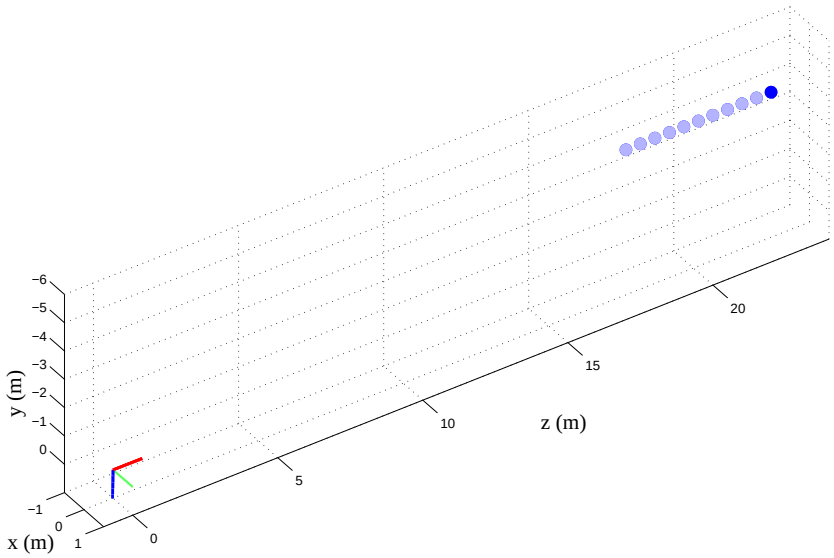

Figure 4.5: A stereo camera is fixed at (0,0,0). One landmark is located at $(1, -5, z)$ with respect to the camera frame and its z coordinate varies in the position estimation problem.

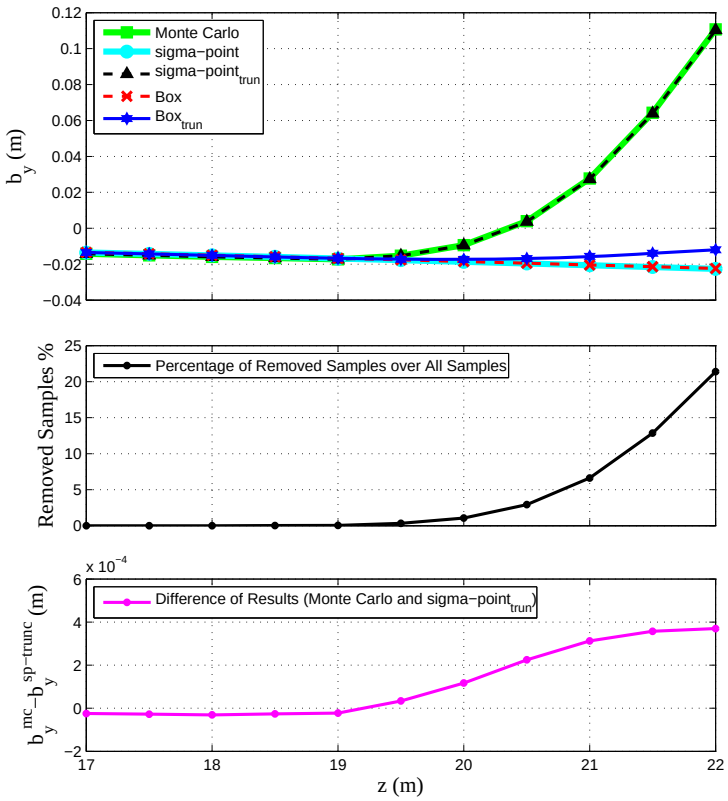

Figure 4.6: Bias estimation (b_y component only) obtained by five methods as a function of landmark position, z (other landmark position coordinates are fixed at $(x, y) = (1, -5)$ in metres) (top); the percentage of samples removed by the truncation procedure (middle); and the difference between b_y obtained based on the Monte Carlo method (our groundtruth) and the truncated sigma-point method (bottom). The measurement noise standard deviation is $\sigma = 0.25$ pixels and the disparity threshold is $d_{\text{th}} = 5$ pixels.

is larger for the landmarks that are located farther from the camera.

Figures 4.7 and 4.8 show the effects of varying the noise, σ, and the disparity threshold, d_{th}. Once more, only the truncated sigma-point method performs well in tracking the Monte Carlo method. The other methods estimate bias in the wrong direction. It is seen from Figure 4.7 that increasing noise results in the growth of the estimated bias. The estimated bias increases 0.22 metres when the noise standard deviation varies from 0.1 pixels to 0.7 pixels.

Additionally, we can observe in Figure 4.8 that the estimated bias increases by 0.09 metres as the disparity threshold increases from 4 pixels to 5.5 pixels. All the methods except for the truncated sigma-point fail to track ground truth in this scenario when the sample truncation begins.

Discussion on Simulation Results

From the simulation results, it can be inferred that without imposing necessary modifications on the sigma-point method, the method is not successful in estimating bias in the position estimation problem. Without modification, the sigma-points are generated based on $\mathbf{p}_{\text{t}}$ and $\mathbf{R}$. The method does not incorporate the effect of the disparity threshold in sampling and thus some of the sigma-points may not pass the disparity threshold test. Whether we truncate bad sigma-points or keep them to be passed through the estimator, these deterministic points are not able to appropriately represent the truncated sampling distribution and it will not estimate bias correctly. Approximating the sampling distribution in a correct way (refitting a Gaussian to the remaining samples) is, in fact, the reason the truncated sigma-point method succeeds in the simulations.

The Box method is an analytical technique that employs the Taylor series expan-

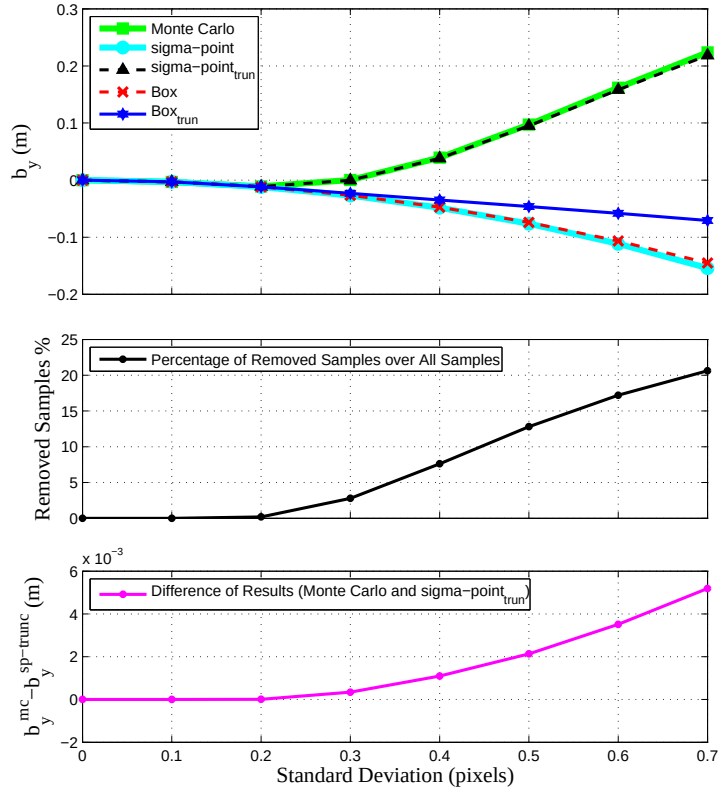

Figure 4.7: Bias estimation (b_y component only) obtained by five methods as a function of measurement noise standard deviation, σ, in pixels (top); the percentage of removed samples over all samples in the truncation procedure (middle); and the difference between b_y obtained based on the Monte Carlo method (our groundtruth) and the truncated sigma-point method (bottom). In our simulations, the landmark is at $(1, -5, 20)$ in metres and the disparity threshold is $d_{\text{th}} = 5$ pixels.

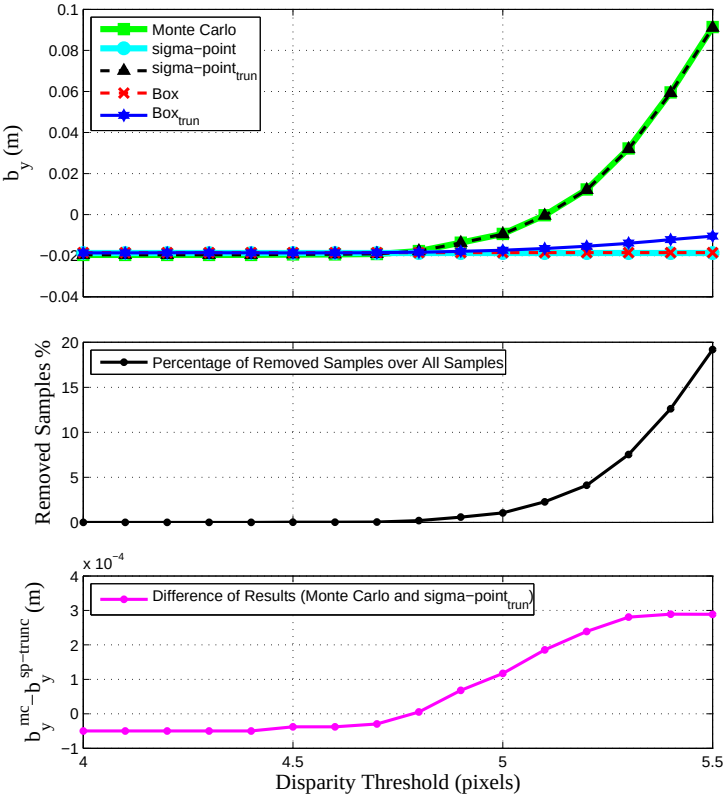

Figure 4.8: Bias estimation (b_y component only) obtained by five methods as a function of disparity threshold, d_{th} (top); the percentage of removed samples over all samples in the truncation procedure (middle); and the difference between b_y obtained based on the Monte Carlo method (our groundtruth) and the truncated sigma-point method (bottom). In our simulations, the landmark is at $(1, -5, 20)$ in metres and the measurement noise standard deviation is $\sigma = 0.25$ pixels.

sion up to second order. Approximating the nonlinear function with its second-order Taylor expansion may be one of the reasons that causes this method to fail. As a result, the bias estimation model cannot represent the modified distribution correctly.

In our analysis, we do not develop the Box method to use higher order approximations; however, we think that doing so might improve its performance. In addition, the Box method approximates bias dependency on the noise up to the quadratic term. This might also be an inappropriate assumption that makes the method to fail. The idea of the truncated Box method is to alter the covariance matrix and the operating point used in the linearization process of the original Box method. Although the modifications improve estimation, the result still does not track ground truth well.

Chapter 5

Visual Odometry Bias Estimation

This chapter is devoted to the bias estimation techniques used for the frame-to-frame VO problem. In Section 5.1, we describe the analytical framework of the bias estimation methods, including the Monte Carlo method, the sigma-point approach, and the Box technique, in two phases.

The first phase, which is described in Subsection 5.1.1, considers the assumption of knowing ground truth in the process of estimating bias and produces bias corrected estimates. In the second phase discussed in Subsection 5.1.2, we eliminate the assumption of knowing the true state and propose the bias estimation techniques which will be applicable to real-world VO problems. We conclude this chapter by Section 5.2 in which we provide the simulation results for bias estimation and show the bias corrected path estimates using different methods.

5.1 Analytical Framework

5.1.1 VO Bias Estimation using Ground Truth

Mathematical Formulation for the Monte Carlo and the Sigma-point Methods in the VO Framework

In this section, we apply our bias estimation techniques to the frame-to-frame VO problem outlined in Chapter 3, assuming that ground truth is available. We consider a simple frame-to-frame scenario where a camera observes some landmarks at time k', moves, and then reobserves the same landmarks at time k; our job is to estimate how the camera moved (and, optionally, the positions of the landmarks).

To produce sigma-points for our problem, we use J landmarks. Each landmark j has an interpretation in image coordinates at time k', $\mathbf{y}_{jk',\mathrm{t}}$, and time k, $\mathbf{y}_{jk,\mathrm{t}}$ given by

$$\mathbf{y}_{jk,\mathrm{t}} = \mathbf{f}(\mathbf{T}_{k,\mathrm{t}}\mathbf{p}_{j,\mathrm{t}}), \qquad \mathbf{y}_{jk',\mathrm{t}} = \mathbf{f}(\mathbf{p}_{j,\mathrm{t}}), \tag{5.1}$$

where $\mathbf{T}_{k,\mathrm{t}}$ and $\mathbf{p}_{j,\mathrm{t}}$ are the true state variables. We stack all these noise-free measurements into one vector as follows:

$$\mathbf{y}_{\mathrm{t}} := \left[\begin{array}{c} \mathbf{y}_{1k'} \\ \vdots \\ \mathbf{y}_{Jk'} \\ \hline \mathbf{y}_{1k} \\ \vdots \\ \mathbf{y}_{Jk} \end{array}\right]_{\mathrm{t}}. \tag{5.2}$$

Then we use this constructed $\mathbf{y}_{\mathrm{t}}$ and equations (4.5) to (4.7) to generate required samples (Monte Carlo samples or sigma-points), $\mathbf{y}_i$, as inputs to the nonlinear VO

estimation algorithm, $\mathbf{h}_{\text{vo}}$ (refer to (3.57)), to produce corresponding estimates $\{\hat{\boldsymbol{\pi}}_k\}_i$ as follows:

$$\{\hat{\boldsymbol{\pi}}_k\}_i = \mathbf{h}_{\text{vo}}(\mathbf{y}_i). \tag{5.3}$$

We obtain each estimate of the transformation matrix, $\{\hat{\mathbf{T}}_k\}_i$, from the corresponding camera pose estimate, $\{\hat{\boldsymbol{\pi}}_k\}_i$, using the expressions (2.4) to (2.10) in Chapter 2. The estimates $\{\hat{\mathbf{T}}_k\}_i$ are used to compute the following weighted sum

$$\hat{\mathbf{T}}'_{k,\text{m}} = \sum_{i=1}^{I} \omega_i \{\hat{\mathbf{T}}_k\}_i, \tag{5.4}$$

where I is the number of Monte Carlo samples or sigma-points and the ω_i were introduced previously in Subsection 4.3. The weighted sum in (5.4) will result in a 4×4 matrix, denoted by $\hat{\mathbf{T}}'_{k,\text{m}}$, that is not in *SE(3)*. To recover the required properties to make (5.4) an element of *SE(3)*, we first partition the result as follows:

$$\hat{\mathbf{T}}'_{k,\text{m}} := \left[\begin{array}{c|c} \mathbf{C}' & \mathbf{d} \\ \hline \mathbf{0}^T & 1 \end{array}\right]_{4\times4}, \tag{5.5}$$

where $\mathbf{C}'$ is a 3×3 matrix and $\mathbf{d}$ is a 3×1 vector. Then, we define a mean estimate to be an element of *SE(3)* denoted by $\hat{\mathbf{T}}_{k,\text{m}}$ as follows:

$$\hat{\mathbf{T}}_{k,\text{m}} := \left[\begin{array}{c|c} \mathbf{C} & \mathbf{d} \\ \hline \mathbf{0}^T & 1 \end{array}\right]_{4\times4}, \tag{5.6}$$

where $\mathbf{C}$ is a 3×3 rotation matrix that satisfies

$$\mathbf{C}^T\mathbf{C} = \mathbf{1}_{3\times3}, \tag{5.7}$$

and is given by

$$\mathbf{C} := \mathbf{C}'(\mathbf{C}'^T\mathbf{C}')^{-\frac{1}{2}}. \tag{5.8}$$

We claim that $\mathbf{C}$ defined as in (5.8) is a rotation matrix. To achieve this, it is sufficient to demonstrate that $\det \mathbf{C} = 1$ and $\mathbf{C}^{-1} = \mathbf{C}^T$. It is obvious that $\det \mathbf{C} = 1$, we only show the second part. For a symmetric invertible matrix A, $(A^T)^{-1/2} = (A^{-1/2})^T$, from which it follows that

$$\begin{aligned}
\mathbf{C}^T\mathbf{C} &= \left((\mathbf{C}'^T\mathbf{C}')^{-\frac{1}{2}}\right)^T \mathbf{C}'^T\mathbf{C}'(\mathbf{C}'^T\mathbf{C}')^{-\frac{1}{2}} \\
&= \left((\mathbf{C}'^T\mathbf{C}')^{-\frac{1}{2}}\right)^T (\mathbf{C}'^T\mathbf{C}')^{\frac{1}{2}} \\
&= (\mathbf{C}'^T\mathbf{C}')^{-\frac{1}{2}}(\mathbf{C}'^T\mathbf{C}')^{\frac{1}{2}} \\
&= I.
\end{aligned}$$

Note that when we obtain $\{\hat{\mathbf{T}}_k\}_i$ from $\{\hat{\boldsymbol{\pi}}_k\}_i$ (computed by (5.3)), we must similarly verify that each $\{\hat{\mathbf{T}}_k\}_i$ is constructed from a rotation matrix that satisfies (5.7) and has a similar format as described in (5.6). Thus, $\{\hat{\mathbf{T}}_k\}_i$ are also elements of *SE(3)*. In general, bias is defined as the difference between the mean value of a parameter estimate and the true value of the parameter. Using the identity $\|\hat{\mathbf{A}} - \mathbf{A}_\mathrm{t}\| = \|\mathbf{A}_\mathrm{t}\|\|\hat{\mathbf{A}}\,\mathbf{A}_\mathrm{t}^{-1} - \mathbf{1}\|$, bias can also be presented by $\hat{\mathbf{A}}\,\mathbf{A}_\mathrm{t}^{-1}$. Knowing that $\mathbf{T}_{k,\mathrm{t}}$ is an invertible matrix (refer to Subsection 2.3.1), the bias in the pose estimate is given by

$$\hat{\mathbf{T}}_{k,\mathrm{bias}} \approx \hat{\mathbf{T}}_{k,\mathrm{m}}\mathbf{T}_{k,\mathrm{t}}^{-1}, \tag{5.9}$$

This is the counterpart to (4.10) for estimating the bias in the $\hat{\mathbf{T}}_k$ matrix. We then convert this 4×4 transformation matrix, $\hat{\mathbf{T}}_{k,\mathrm{bias}}$, to a 6×1 camera pose vector, $\hat{\boldsymbol{\pi}}_{k,\mathrm{bias}}$, that contains information concerning translation and rotation of the camera using (2.4) to (2.10). This facilitates easier plotting of the six degrees of freedom in the pose bias. If we break $\hat{\boldsymbol{\pi}}_{k,\mathrm{bias}}$ down into its components we have

$$\hat{\boldsymbol{\pi}}_{k,\text{bias}} = \begin{bmatrix} \hat{\boldsymbol{\rho}} \\ \hat{\boldsymbol{\phi}} \end{bmatrix}_{k,\text{bias}}, \quad \hat{\boldsymbol{\rho}}_{k,\text{bias}} = \begin{bmatrix} \hat{\rho}_x \\ \hat{\rho}_y \\ \hat{\rho}_z \end{bmatrix}_{k,\text{bias}}. \tag{5.10}$$

The estimated bias obtained by (5.9) is used to correct for bias in the VO estimates. We now remove the estimated bias from the original VO estimate $\hat{\mathbf{T}}_k$ $(= \mathbf{h}_{\text{vo}}(\mathbf{y}_{\text{meas}}))$ to provide a modified version of the estimate denoted by $\hat{\mathbf{T}}_{k,\text{mod}}$

$$\hat{\mathbf{T}}_{k,\text{mod}} = (\hat{\mathbf{T}}_{k,\text{bias}})^{-1}\hat{\mathbf{T}}_k. \tag{5.11}$$

While this approach does a reasonable job of estimating bias, it requires knowledge of the ground truth trajectory, and therefore is not suitable for online use. To prove that the bias is greatly reduced in the modified estimate, $\hat{\mathbf{T}}_{k,\text{mod}}$, we compute bias for $\hat{\mathbf{T}}_{k,\text{mod}}$. Assume we obtain a sufficiently large number of estimates, $\{\hat{\mathbf{T}}_k\}_{i'}$. We produce the corresponding modified estimates, $\{\hat{\mathbf{T}}_{k,\text{mod}}\}_{i'}$, using (5.9) and (5.11). The bias in $\hat{\mathbf{T}}_{k,\text{mod}}$ is given by

$$\begin{aligned}
&= \sum_{i'=1}^{I'} \omega_{i'} \{\hat{\mathbf{T}}_{k,\text{mod}}\}_{i'} \mathbf{T}_{k,\text{t}}^{-1} \\
&= \sum_{i'=1}^{I'} \omega_{i'} \{(\hat{\mathbf{T}}_{k,\text{bias}})^{-1}\hat{\mathbf{T}}_k\}_{i'} \mathbf{T}_{k,\text{t}}^{-1} \\
&= \sum_{i'=1}^{I'} \omega_{i'} \{(\hat{\mathbf{T}}_{k,\text{m}}\mathbf{T}_{k,\text{t}}^{-1})^{-1}\hat{\mathbf{T}}_k\}_{i'} \mathbf{T}_{k,\text{t}}^{-1} \\
&\approx \sum_{i'=1}^{I'} \omega_{i'} \{(\sum_{i=1}^{I} \omega_i \{\hat{\mathbf{T}}_k\}_i \mathbf{T}_{k,\text{t}}^{-1})^{-1}\hat{\mathbf{T}}_k\}_{i'} \mathbf{T}_{k,\text{t}}^{-1} \\
&\approx (\sum_{i=1}^{I} \omega_i \{\hat{\mathbf{T}}_k\}_i \mathbf{T}_{k,\text{t}}^{-1})^{-1} \sum_{i'=1}^{I'} \omega_{i'} \{\hat{\mathbf{T}}_k\}_{i'} \mathbf{T}_{k,\text{t}}^{-1} \\
&\approx \mathbf{I}_{4\times 4}
\end{aligned} \tag{5.12}$$

where $\mathbf{1}_{4\times4}$ is an identity matrix and the result shows the average of modified estimates contains no bias. Note that in the real VO problems only one estimate is produced and corrected at one time step; therefore, we must look at the outcome of the bias correction algorithm applied to a long distance of travel. We assume that measurement noise standard deviation remains the same at each time step and landmarks are randomly distributed on the ground. We show by simulations (refer to 5.2.2) that the effect of bias correction algorithm becomes apparent after traversing a long path which is computed by compounding the estimates that are bias-corrected at each time step. The methods described in this section will be utilized in Section 5.2 for bias estimation simulations.

Mathematical Formulation for Box's Method in the VO Framework

In Chapter 4 we introduced Box's expression for quantifying bias in a generic nonlinear estimation problem (see Subsection 4.3). The expression was used for a stereo-triangulation problem in Subsection 4.5. In this subsection, using [6] we show the derivation procedure to produce an expression of bias estimate for the VO problem by Box's method. The expression will be used for simulations described in Section 5.1.2. We assume that $\delta\hat{\boldsymbol{\xi}}$ is defined as the difference between our estimate, $\hat{\mathbf{x}}$, and the true state, $\mathbf{x}_{\mathrm{t}}$, for a specific realization of the sensor noise, $\mathbf{n}$. Then the following expectation over all realizations of sensor noise is our bias estimate,

$$E[\hat{\mathbf{x}}] - \mathbf{x}_{\mathrm{t}} \;=\; E[\hat{\mathbf{x}} - \mathbf{x}_{\mathrm{t}}] = E[\,\delta\hat{\boldsymbol{\xi}}\,]. \tag{5.13}$$

We rederive the bias expression proposed by Box [14] for the VO problem. Our measurement model is such that

$$\mathbf{y}_{jk} = \mathbf{h}_{jk}(\mathbf{x}_{\mathrm{t}}) + \mathbf{n}_{jk},$$

where $\mathbf{n}_{jk} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{jk})$ is zero-mean additive Gaussian sensor noise introduced in (3.17) and $\mathbf{h}_{jk}(\cdot)$ is defined in (3.32). For simplicity, we stack all the sensor noise variables into a column, $\mathbf{n}$, such that $\mathbf{n}_{jk} = \mathbf{P}_{\mathbf{n}jk}\,\mathbf{n}$ and $\mathbf{P}_{\mathbf{n}jk}$ is the projection matrix defined as:

$$\mathbf{P}_{\mathbf{n}jk} := \left[\begin{array}{ccccc|ccccc} \mathbf{0}_{4\times4} & \cdots & \mathbf{0}_{4\times4} & \cdots & \mathbf{0}_{4\times4} & \mathbf{0}_{4\times4} & \cdots & \mathbf{1}_{4\times4} & \cdots & \mathbf{0}_{4\times4} \\ 1 & \cdots & j & \cdots & J & 1 & \cdots & j & \cdots & J \end{array}\right] \tag{5.14}$$

In the Box's bias estimation method, a second-order Taylor expansion of $\mathbf{h}_{jk}$ and a first-order expansion of $\mathbf{H}_{jk}$ will be utilized. Thus, we have the following linearized expressions:

$$\begin{aligned} \mathbf{h}_{jk}(\hat{\mathbf{x}}_{jk}) = \mathbf{h}_{jk}(\mathbf{x}_{jk,\mathrm{t}} + \delta\hat{\boldsymbol{\xi}}_{jk}) \;&\approx\; \mathbf{h}_{jk}(\mathbf{x}_{jk,\mathrm{t}}) + \mathbf{H}_{jk}(\mathbf{x}_{jk,\mathrm{t}})\,\mathbf{P}_{\mathbf{n}jk}\,\delta\hat{\boldsymbol{\xi}} \\ &\quad + \frac{1}{2}\sum_i \mathbf{1}_i\,\delta\hat{\boldsymbol{\xi}}^T\,\mathbf{P}_{\mathbf{n}jk}{}^T\boldsymbol{\mathcal{H}}_{ijk}(\mathbf{x}_{jk,\mathrm{t}})\,\mathbf{P}_{\mathbf{n}jk}\,\delta\hat{\boldsymbol{\xi}}, \end{aligned} \tag{5.15}$$

$$\mathbf{H}_{jk}(\hat{\mathbf{x}}_{jk}) = \mathbf{H}_{jk}(\mathbf{x}_{jk,\mathrm{t}} + \delta\hat{\boldsymbol{\xi}}_{jk}) \;\approx\; \mathbf{H}_{jk}(\mathbf{x}_{jk,\mathrm{t}}) + \sum_i \mathbf{1}_i\,\mathbf{P}_{\mathbf{n}jk}{}^T\,\delta\hat{\boldsymbol{\xi}}^T\boldsymbol{\mathcal{H}}_{ijk}(\mathbf{x}_{jk,\mathrm{t}}), \tag{5.16}$$

where $\delta\hat{\boldsymbol{\xi}}_{jk} = \mathbf{P}_{\mathbf{n}jk}\delta\hat{\boldsymbol{\xi}}$. Note that we can also state the same equations for the previous time step k' as above. Now we need to find an estimate, $\hat{\mathbf{x}}$, that minimizes the objective function given by (3.21). Therefore the optimality criterion gives

$$\left.\frac{\partial^T Q(\mathbf{x}_\mathrm{t})}{\partial \mathbf{x}_\mathrm{t}}\right|_{\hat{\mathbf{x}}} = \mathbf{0},$$

or equivalently

$$\sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}{}^T\mathbf{H}_{jl}(\hat{\mathbf{x}})^T\mathbf{R}_{jl}^{-1}\left(\mathbf{y}_{jl} - \mathbf{h}_{jl}(\hat{\mathbf{x}})\right) = \mathbf{0},$$

or

$$\sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \left(\mathbf{H}_{jl}(\mathbf{x}_\mathrm{t}) + \sum_i \mathbf{1}_i \, \delta\hat{\boldsymbol{\xi}}^T \mathbf{P}_{\mathbf{n}jl}^T \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t}) \right)^T \mathbf{R}_{jl}^{-1} \times$$
$$\left(\underbrace{\mathbf{y}_{jl} - \mathbf{h}_{jl}(\mathbf{x}_\mathrm{t})}_{\mathbf{n}_{jl}} - \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t}) \mathbf{P}_{\mathbf{n}jl} \, \delta\hat{\boldsymbol{\xi}} - \frac{1}{2} \sum_i \mathbf{1}_i \, \delta\hat{\boldsymbol{\xi}}^T \mathbf{P}_{\mathbf{n}jl}^T \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t}) \mathbf{P}_{\mathbf{n}jl} \, \delta\hat{\boldsymbol{\xi}} \right) \approx \mathbf{0},$$

after substituting (5.15) and (5.16). We will assume that $\delta\hat{\boldsymbol{\xi}}$ depends on $\mathbf{n}$ up to the quadratic term:

$$\delta\hat{\boldsymbol{\xi}} = \mathbf{A}(\mathbf{x}_\mathrm{t}) \, \mathbf{n} + \mathbf{b}(\mathbf{n}), \tag{5.17}$$

where $\mathbf{A}$ is an unknown coefficient matrix and $\mathbf{b}(\mathbf{n})$ is an unknown vector that is a quadratic function of $\mathbf{n}$. Substituting (5.17) we have that

$$\sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \left(\mathbf{H}_{jl}(\mathbf{x}_\mathrm{t}) + \sum_i \mathbf{1}_i \, \left(\mathbf{A}(\mathbf{x}_\mathrm{t}) \, \mathbf{n} + \mathbf{b}(\mathbf{n}) \right)^T \mathbf{P}_{\mathbf{n}jl}^T \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t}) \right)^T \mathbf{R}_{jl}^{-1} \times$$
$$\left(\mathbf{P}_{\mathbf{n}jl} \, \mathbf{n} - \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t}) \mathbf{P}_{\mathbf{n}jl} \left(\mathbf{A}(\mathbf{x}_\mathrm{t}) \, \mathbf{n} + \mathbf{b}(\mathbf{n}) \right) \right.$$
$$\left. - \frac{1}{2} \sum_i \mathbf{1}_i \left(\mathbf{A}(\mathbf{x}_\mathrm{t}) \, \mathbf{n} + \mathbf{b}(\mathbf{n}) \right)^T \mathbf{P}_{\mathbf{n}jl}^T \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t}) \mathbf{P}_{\mathbf{n}jl} \left(\mathbf{A}(\mathbf{x}_\mathrm{t}) \, \mathbf{n} + \mathbf{b}(\mathbf{n}) \right) \right) \approx \mathbf{0}.$$

Multiplying out and keeping terms up to quadratic in $\mathbf{n}$ we have

$$\underbrace{\sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T \mathbf{R}_{jl}^{-1} \left(\mathbf{P}_{\mathbf{n}jl} - \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t}) \mathbf{P}_{\mathbf{n}jl} \mathbf{A}(\mathbf{x}_\mathrm{t}) \right) \mathbf{n}}_{\mathbf{L}\,\mathbf{n} \quad \text{(linear in } \mathbf{n}\text{)}}$$
$$- \underbrace{\sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T \mathbf{R}_{jl}^{-1} \left(\mathbf{H}_{jl}(\mathbf{x}_\mathrm{t}) \mathbf{P}_{\mathbf{n}jl} \mathbf{b}(\mathbf{n}) + \frac{1}{2} \sum_i \mathbf{1}_i \underbrace{\left(\mathbf{P}_{\mathbf{n}jl} \mathbf{A}(\mathbf{x}_\mathrm{t}) \mathbf{n} \right)^T \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t}) \mathbf{P}_{\mathbf{n}jl} \mathbf{A}(\mathbf{x}_\mathrm{t}) \mathbf{n}}_{\text{scalar}} \right)}_{\mathbf{q}_1(\mathbf{n}) \quad \text{(quadratic in } \mathbf{n}\text{)}}$$
$$+ \underbrace{\sum_{\substack{i,j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t})^T \mathbf{P}_{\mathbf{n}jl} \mathbf{A}(\mathbf{x}_\mathrm{t}) \, \mathbf{n} \underbrace{\mathbf{1}_i^T \mathbf{R}_{jl}^{-1} \left(\mathbf{P}_{\mathbf{n}jl} - \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t}) \mathbf{P}_{\mathbf{n}jl} \mathbf{A}(\mathbf{x}_\mathrm{t}) \right) \mathbf{n}}_{\text{scalar}}}_{\mathbf{q}_2(\mathbf{n}) \quad \text{(quadratic in } \mathbf{n}\text{)}} \approx \mathbf{0}.$$

In order to set the above expression identically zero (up to second order in $\mathbf{n}$),

$$\mathbf{L}\,\mathbf{n} + \mathbf{q}_1(\mathbf{n}) + \mathbf{q}_2(\mathbf{n}) = \mathbf{0},$$

it then follows that $\mathbf{L} = \mathbf{0}$, since considering the case of the opposing sign of $\mathbf{n}$ gives

$$-\mathbf{L}\,\mathbf{n} + \mathbf{q}_1(-\mathbf{n}) + \mathbf{q}_2(-\mathbf{n}) = \mathbf{0},$$

using the fact that $\mathbf{q}_1(-\mathbf{n}) = \mathbf{q}_1(\mathbf{n})$ and $\mathbf{q}_2(-\mathbf{n}) = \mathbf{q}_2(\mathbf{n})$ due to the quadratic nature of these terms, it then follows that $\mathbf{L}\,\mathbf{n} = \mathbf{0}$ and since $\mathbf{n}$ can take on any value, the desired result is obtained, that is, $\mathbf{L} = \mathbf{0}$. We now have

$$\mathbf{A}(\mathbf{x}_\mathrm{t}) = \mathbf{W}(\mathbf{x}_\mathrm{t})^{-1} \sum_{\substack{j \\ l \in \{k', k\}}} \mathbf{P}_{\mathbf{n}jl}^T \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T \mathbf{R}_{jl}^{-1} \mathbf{P}_{\mathbf{n}jl},$$

where

$$\mathbf{W}(\mathbf{x}_\mathrm{t}) := \sum_{\substack{j \\ l \in \{k', k\}}} \mathbf{P}_{\mathbf{n}jl}^T \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T \mathbf{R}_{jl}^{-1} \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t}) \mathbf{P}_{\mathbf{n}jl}.$$

Choosing this value for $\mathbf{A}(\mathbf{x}_\mathrm{t})$ and taking the expectation (over all values of $\mathbf{n}$) we are left with

$$E\left[\mathbf{q}_1(\mathbf{n})\right] + E\left[\mathbf{q}_2(\mathbf{n})\right] = \mathbf{0}.$$

Fortunately, it turns out that $E\left[\mathbf{q}_2(\mathbf{n})\right] = \mathbf{0}$. To see this, we need two identities:

$$\begin{aligned} \mathbf{A}(\mathbf{x}_\mathrm{t})\mathbf{R}\mathbf{A}(\mathbf{x}_\mathrm{t})^T &\equiv \mathbf{W}(\mathbf{x}_\mathrm{t})^{-1} \\ \mathbf{A}(\mathbf{x}_\mathrm{t})\mathbf{R}\mathbf{P}_{\mathbf{n}jk}^T &\equiv \mathbf{W}(\mathbf{x}_\mathrm{t})^{-1} \mathbf{P}_{\mathbf{n}jk}^T \mathbf{H}_{jk}(\mathbf{x}_\mathrm{t}) \end{aligned}$$

where $\mathbf{R}_{jk} = \mathbf{P}_{\mathbf{n}jk}\,\mathbf{R}\,\mathbf{P}_{\mathbf{n}jk}^T$. We then have

$$
\begin{aligned}
E\left[\mathbf{q}_2(\mathbf{n})\right] &= E\left[\sum_{\substack{i,j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t})^T \mathbf{P}_{\mathbf{n}jl}\mathbf{A}(\mathbf{x}_\mathrm{t})\,\mathbf{n}\,\mathbf{1}_i^T \mathbf{R}_{jl}^{-1}\left(\mathbf{P}_{\mathbf{n}jl} - \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})\mathbf{P}_{\mathbf{n}jl}\mathbf{A}(\mathbf{x}_\mathrm{t})\right)\,\mathbf{n}\right] \\
&= \sum_{\substack{i,j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t})^T \mathbf{P}_{\mathbf{n}jl}\mathbf{A}(\mathbf{x}_\mathrm{t})\,\underbrace{E\left[\mathbf{n}\,\mathbf{n}^T\right]}_{\mathbf{R}}\left(\mathbf{P}_{\mathbf{n}jl}^T - \mathbf{A}(\mathbf{x}_\mathrm{t})^T\mathbf{P}_{\mathbf{n}jl}^T\mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T\right)\mathbf{R}_{jl}^{-1}\,\mathbf{1}_i \\
&= \sum_{\substack{i,j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t})^T(\mathbf{P}_{\mathbf{n}jl}\underbrace{\mathbf{A}(\mathbf{x}_\mathrm{t})\mathbf{R}\mathbf{P}_{\mathbf{n}jl}^T}_{\mathbf{W}(\mathbf{x}_\mathrm{t})^{-1}\mathbf{P}_{\mathbf{n}jl}^T\mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})} - \mathbf{P}_{\mathbf{n}jl}\underbrace{\mathbf{A}(\mathbf{x}_\mathrm{t})\mathbf{R}\mathbf{A}(\mathbf{x}_\mathrm{t})^T}_{\mathbf{W}(\mathbf{x}_\mathrm{t})^{-1}}\mathbf{P}_{\mathbf{n}jl}^T\mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T)\mathbf{R}_{jl}^{-1}\mathbf{1}_i \\
&= \mathbf{0},
\end{aligned}
$$

where we have made use of the above identities. As a result, the remaining term of (5.1.1) is zero, that is,

$$
E\left[\mathbf{q}_1(\mathbf{n})\right] = \mathbf{0}, \tag{5.18}
$$

or equivalently

$$
\begin{aligned}
E[\mathbf{b}(\mathbf{n})] &= \mathbf{K}\sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T \mathbf{R}_{jl}^{-1}\sum_i \mathbf{1}_i\, E\left[\,\mathbf{n}^T\,\mathbf{A}(\mathbf{x}_\mathrm{t})^T\mathbf{P}_{\mathbf{n}jl}^T\boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t})\mathbf{P}_{\mathbf{n}jl}\,\mathbf{A}(\mathbf{x}_\mathrm{t})\,\mathbf{n}\,\right] \\
&= \mathbf{K}\sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T \mathbf{R}_{jl}^{-1}\sum_i \mathbf{1}_i\, E\left[\mathrm{tr}\left(\mathbf{P}_{\mathbf{n}jl}^T\boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t})\mathbf{P}_{\mathbf{n}jl}\,\mathbf{A}(\mathbf{x}_\mathrm{t})\,\mathbf{n}\,\mathbf{n}^T\,\mathbf{A}(\mathbf{x}_\mathrm{t})^T\right)\right] \\
&= \mathbf{K}\sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T \mathbf{R}_{jl}^{-1}\sum_i \mathbf{1}_i\,\mathrm{tr}\big(\mathbf{P}_{\mathbf{n}jl}^T\boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t})\mathbf{P}_{\mathbf{n}jl}\underbrace{\mathbf{A}(\mathbf{x}_\mathrm{t})\,\underbrace{E\left[\mathbf{n}\,\mathbf{n}^T\right]}_{\mathbf{R}}\,\mathbf{A}(\mathbf{x}_\mathrm{t})^T}_{\mathbf{W}(\mathbf{x}_\mathrm{t})^{-1}}\big) \\
&= \mathbf{K}\sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^T \mathbf{H}_{jl}(\mathbf{x}_\mathrm{t})^T \mathbf{R}_{jl}^{-1}\sum_i \mathbf{1}_i\,\mathrm{tr}\big(\mathbf{P}_{\mathbf{n}jl}^T\boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_\mathrm{t})\mathbf{P}\mathbf{P}_{\mathbf{n}jl}\,\mathbf{W}(\mathbf{x}_\mathrm{t})^{-1}\big),
\end{aligned}
$$

where $\mathrm{tr}(\cdot)$ indicates the trace of a matrix and $\mathbf{K} = \frac{-\mathbf{W}(\mathbf{x}_\mathrm{t})^{-1}}{2}$. Employing again (5.17), we observe that

$$E[\delta\hat{\boldsymbol{\xi}}] = \mathbf{A}(\mathbf{x}_{\mathrm{t}}) \underbrace{E[\mathbf{n}]}_{\mathbf{0}} + E[\mathbf{b}(\mathbf{n})],$$

Finally, the bias estimation expression is given by

$$E[\delta\hat{\boldsymbol{\xi}}] \approx \mathbf{K} \sum_{\substack{j \\ l\in\{k',k\}}} \mathbf{P}_{\mathbf{n}jl}^{T} \mathbf{H}_{jl}(\mathbf{x}_{\mathrm{t}})^{T} \mathbf{R}_{jl}^{-1} \sum_{i} \mathbf{1}_i \,\mathrm{tr}\big(\mathbf{P}_{\mathbf{n}jl}^{T} \boldsymbol{\mathcal{H}}_{ijl}(\mathbf{x}_{\mathrm{t}}) \mathbf{P}_{\mathbf{n}jl}\, \mathbf{W}(\mathbf{x}_{\mathrm{t}})^{-1}\big). \tag{5.19}$$

For the truncated Box method, we utilize truncated Gaussian statistics in (4.14) and (4.15) and our nonlinear estimator $\mathbf{h}_{\mathrm{vo}}(\cdot)$ defined in (3.57) to produce the appropriate operating point for evaluation of above matrices. Instead of using the true state, $\mathbf{x}_{\mathrm{t}}$, the matrices in (5.19) are evaluated at $\hat{\mathbf{x}}^{\mathrm{mc}}_{\mathrm{m,trun}}$ given by

$$\hat{\mathbf{x}}^{\mathrm{mc}}_{\mathrm{m,trun}} \;=\; \mathbf{h}_{\mathrm{vo}}(\mathbf{y}^{\mathrm{mc}}_{\mathrm{m,trun}}), \tag{5.20}$$

where $\mathbf{y}^{\mathrm{mc}}_{\mathrm{m,trun}}$ is obtained from (4.14). Moreover, the covariance matrix $\mathbf{R}$ used in the expression (5.19) is substituted by $\mathbf{R}^{\mathrm{mc}}_{\mathrm{trun}}$ calculated from (4.15).

The first six elements of the bias estimate vector, $E[\delta\hat{\boldsymbol{\xi}}]$, in (5.19) form the 6×1 vector $\hat{\boldsymbol{\pi}}_{k,\mathrm{bias}}$ which describes the bias estimate in the camera pose. We obtain the transformation matrix corresponding to $\hat{\boldsymbol{\pi}}_{k,\mathrm{bias}}$ using (2.4) to (2.10) and denote it by $\hat{\mathbf{T}}_{k,\mathrm{bias}}$. Using (5.11), the bias is removed from the estimate $\hat{\mathbf{T}}_k$ to produce $\hat{\mathbf{T}}_{k,\mathrm{mod}}$. This method will be utilized in 5.2.1 for the bias estimation simulations.

5.1.2 VO Bias Estimation Without Ground Truth Knowledge

In real-world VO problems, the assumption of knowing ground truth in our bias estimation algorithm is highly restrictive and unrealistic. In this section, we propose a bias estimation method using only measurements. In statistics, the *bootstrap* technique is used to estimate statistical properties of a model [19] based only on random

samples. This technique was first introduced by Efron [31] and developed by Efron and Gong [34], Efron and Tibshirani [35], Diaconis and Efron [28], and the monograph of Efron [32]. The method has been used in regression analysis for calculating bias and the standard error in estimates. For instance, Kothari and Shanken [59] employ the bootstrapping procedures to the regression analysis associated with the expected market returns and their determinants.

Roughly speaking, this method uses the random samples to produce an estimate $\hat{\theta}$ of an unknown parameter θ_{t}. In addition, the random samples are used to construct an empirical distribution, from which more random samples, called bootstrap samples, are drawn. These bootstrap samples are used to estimate statistical properties of $\hat{\theta}$. For example, they can be used to produce an estimate of θ_{t}, called the bootstrap estimate and denoted by Θ^{bs}. If we draw another random sample and repeat the above process, we can generate another bootstrap estimate. After a sufficient number of bootstrap estimates are produced, we can take the average of these bootstrap estimates, $\Theta^{\mathrm{bs}}_{\mathrm{m}}$, and use it to define the bootstrap estimate of bias for the parameter θ_{t} as [19, 50]

$$\hat{b} \approx \Theta^{\mathrm{bs}}_{\mathrm{m}} - \hat{\theta} \tag{5.21}$$

The main result of the bootstrap is that when we obtain $\Theta^{\mathrm{bs}}_{\mathrm{m}}$ by taking the average of a large number of bootstrap estimates, Θ^{bs}, we expect the values $\Theta^{\mathrm{bs}}_{\mathrm{m}} - \hat{\theta}$ and $E[\hat{\theta}] - \theta_{\mathrm{t}}$ to be nearly the same since the bootstrap distribution of $\Theta^{\mathrm{bs}} - \hat{\theta}$ should approximate the sampling distribution of $\hat{\theta} - \theta_{\mathrm{t}}$ sufficiently well. Note that $\hat{\theta}$ is fixed in the expression $\Theta^{\mathrm{bs}} - \hat{\theta}$ and only Θ^{bs} varies as we generate bootstrap estimates. While we cannot resample in VO problems as in the usual bootstrap application, we borrow the bootstrap idea of subtracting the estimate from the average of many sample

estimates to estimate bias. For example in the truncated sigma-point method, we use sigma-points obtained by truncated Monte Carlo statistics instead of using bootstrap samples drawn from an empirical distribution derived from a random sample. We now describe in detail our approach.

Mathematical Formulation for the Monte Carlo and Sigma-point Methods in the VO Framework

Suppose that we have obtained a set of actual measurements, $\mathbf{y}_{\text{meas}}$, from the stereo camera. We use our motion estimation algorithm defined in (3.57) to determine the estimate, $\{\hat{\mathbf{T}}_k, \hat{\mathbf{p}}_j\}$. In order to draw more Monte Carlo samples or sigma-points, we replace ground truth, $\mathbf{y}_{\text{t}}$, by the actual measurements, $\mathbf{y}_{\text{meas}}$, and use expressions (4.5) to (4.7). We denote these samples by $\mathbf{y}_i^*$. Henceforth, the notation $(^*)$ demonstrates a quantity is generated based on the actual measurements and thus the assumption of knowing ground truth is eliminated from the analysis.

The process of producing Monte Carlo samples or sigma-points based on the actual measurements is analogous to the resampling step in the bootstrap. The samples $\mathbf{y}_i^*$ are passed through our estimator, $\mathbf{h}_{\text{vo}}(\cdot)$, and this results in a set of estimates, $\{\hat{\mathbf{T}}_k^*\}_i$, that are used to form the following weighted sum:

$$\hat{\mathbf{T}}'^*_{k,\text{m}} = \sum_{i=1}^{I} \omega_i \{\hat{\mathbf{T}}_k^*\}_i \tag{5.22}$$

To make $\hat{\mathbf{T}}'^*_{k,\text{m}}$ an element of *SE(3)* we use (5.6) to produce $\hat{\mathbf{T}}^*_{k,\text{m}}$. Finally, we utilize the bootstrap idea in estimating bias that is illustrated in (5.21) to calculate the bias estimate, $\hat{\mathbf{T}}^*_{k,\text{bias}}$, in our VO problem as follows:

$$\hat{\mathbf{T}}^*_{k,\text{bias}} \approx \hat{\mathbf{T}}^*_{k,\text{m}} \hat{\mathbf{T}}_k^{-1}. \tag{5.23}$$

Note that ground truth knowledge is not required for estimating bias by (5.23), whereas the expression (5.9) depends on the knowledge of the true pose, $\mathbf{T}_{k,\mathrm{t}}$. After calculating the bias estimate from (5.23), we will modify the original estimates in the way that the compounded modified estimate provides a corrected path estimate in our problem. The original state estimate, $\hat{\mathbf{T}}_k$, is modified by the estimated bias as follows:

$$\hat{\mathbf{T}}^*_{k,\mathrm{mod}} \quad = \quad (\hat{\mathbf{T}}^*_{k,\mathrm{bias}})^{-1}\hat{\mathbf{T}}_k. \tag{5.24}$$

In 5.2.2, we show that for a sufficiently long distance of travel bias is greatly reduced in the compounded camera path obtained by these modified estimates.

Mathematical Formulation for Box's Method in the VO Framework

To compute bias without using ground truth, we still use the same expression described in (5.19); however, the matrices are not evaluated at $\mathbf{x}_{\mathrm{t}}$. Denote the Monte carlo samples which are generated using the actual measurements and passed the disparity threshold test by $\{\mathbf{y}^{*,\mathrm{mc}}_{\mathrm{trun}}\}_i$.

Then, we pass the average of these samples that is represented by $\mathbf{y}^{*,\mathrm{mc}}_{\mathrm{m,trun}}$ through the nonlinear estimator to obtain $\hat{\mathbf{x}}^{*,\mathrm{mc}}_{\mathrm{m,trun}}$

$$\hat{\mathbf{x}}^{*,\mathrm{mc}}_{\mathrm{m,trun}} = \mathbf{h}_{\mathrm{vo}}(\mathbf{y}^{*,\mathrm{mc}}_{\mathrm{m,trun}}). \tag{5.25}$$

Thus, to eliminate the use of ground truth in estimating bias, the matrices in the bias estimate expression, $E[\delta\hat{\boldsymbol{\xi}}^*]$, are evaluated at $\hat{\mathbf{x}}^{*,\mathrm{mc}}_{\mathrm{m,trun}}$. Finally, $\hat{\mathbf{T}}^*_{k,\mathrm{bias}}$ will be obtained from $E[\delta\hat{\boldsymbol{\xi}}^*]$. The bias is removed from the estimate to produce $\hat{\mathbf{T}}^*_{k,\mathrm{mod}}$ using (5.24). In 5.2.2, we utilize this method in simulations to estimate bias.

5.2 Simulations

5.2.1 Simulation Results for Change of Bias by Varying System Parameters

Given that the truncated sigma-point bias estimation was the only algorithm that performed well compared to the brute-force Monte Carlo approach on the single-landmark position estimation problem of Section 4.5, we only use this method for the following simulations.

We performed our simulations for the asymmetric landmark distribution case shown in Figure 4.2 and observed the change in the bias estimate by varying (i) distance of landmarks from camera (Figure 5.1.a), (ii) the standard deviation of the measurement noise (Figure 5.1.b), and (iii) the disparity threshold (Figure 5.1.c). Estimated biases in the direction of $x-$, $y-$, and $z-$ axes that are referred to as $\hat{\rho}_{x,\text{bias}}$, $\hat{\rho}_{y,\text{bias}}$, and $\hat{\rho}_{z,\text{bias}}$ and the norm of the orientation vector, $||\hat{\boldsymbol{\phi}}_{\text{bias}}||$ are shown.

These plots show that there is a small, but measurable bias in both translation ($\hat{\rho}_{y,\text{bias}}$ and $\hat{\rho}_{z,\text{bias}}$) and rotation for the VO problem and this bias increases with landmark distance, measurement noise covariance, and disparity threshold. For instance, in the case that the position of the closest landmark to the camera and the disparity threshold were fixed at 10 m and 4 pixels respectively, we increased the standard deviation from 0.25 pixels to 0.75 pixels. The results from Figure 5.1.b show that $\hat{\rho}_{y,\text{bias}}$ increased from 2×10^{-4} m to 2×10^{-3} m, and $\hat{\rho}_{z,\text{bias}}$ increased from 4×10^{-3} m to 3×10^{-2} m. Also, $||\hat{\boldsymbol{\phi}}_{\text{bias}}||$ increased from 3×10^{-5} rad ($= 0.002^o$) to 2×10^{-4} rad ($= 0.01^o$).

We used the asymmetric configuration of landmarks shown in Figure 4.2 and estimated the camera path by the Monte Carlo approach and the truncated sigma-

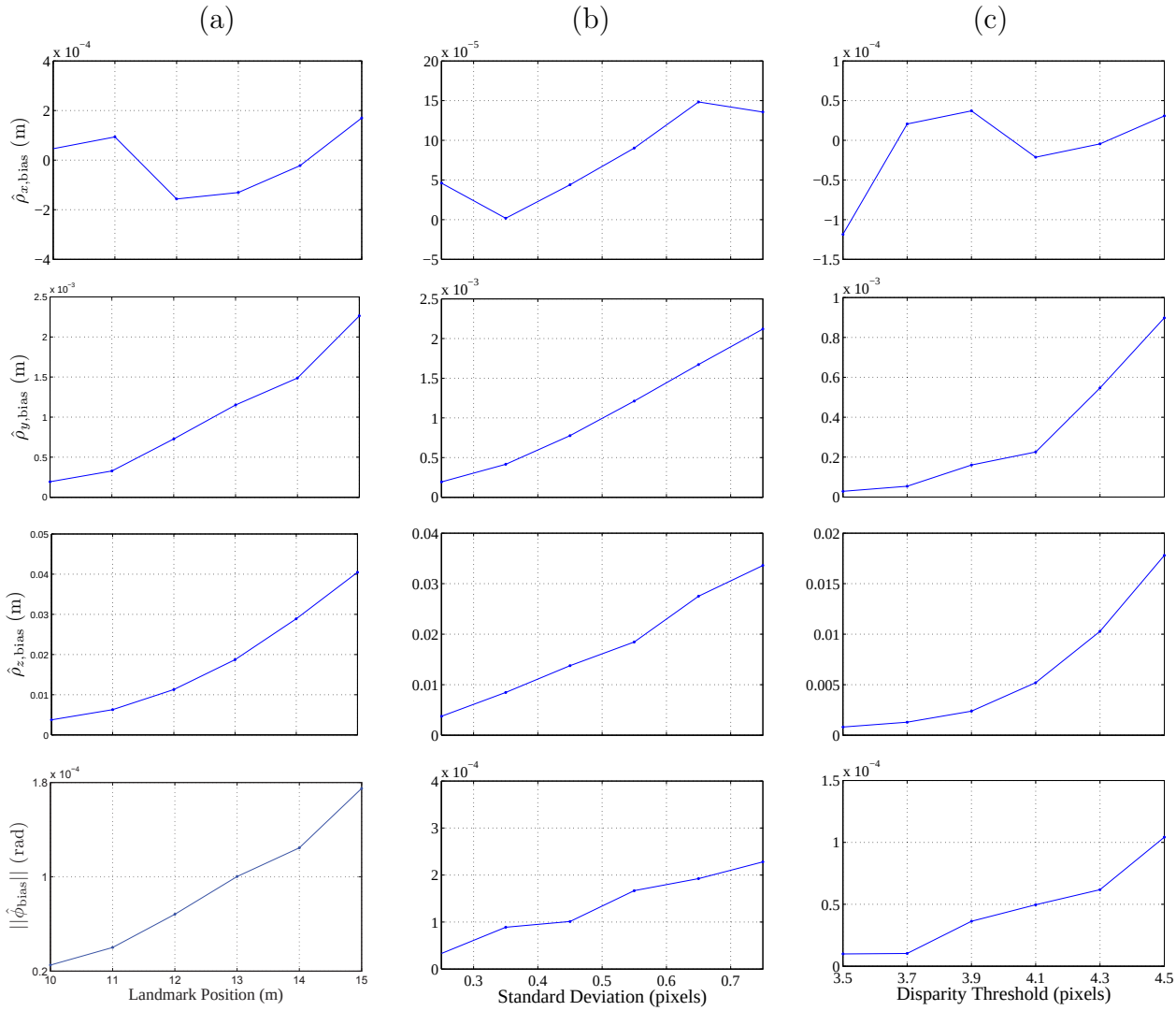

Figure 5.1: (a) Translation bias in the $x-$, $y-$, and $z-$ axes direction, and the norm of the orientation bias versus the position of the closest landmark to the camera are shown. The standard deviation and the disparity threshold are fixed at 0.25 pixels and 4 pixels respectively. (b) Translation bias in the $x-$, $y-$, and $z-$ axes direction, and the norm of the orientation bias versus the standard deviation are indicated. The position of the closest landmark to the camera and the disparity threshold are fixed at 10 metres and 4 pixels respectively. (c) Translation bias in the $x-$, $y-$, and $z-$ axes direction, and the norm of the orientation bias versus the disparity threshold are indicated. The position of the closest landmark to the camera and the standard deviation are fixed at 10 metres and 0.25 pixels respectively. Simulations are performed for one time step based on the truncated sigma-point method.

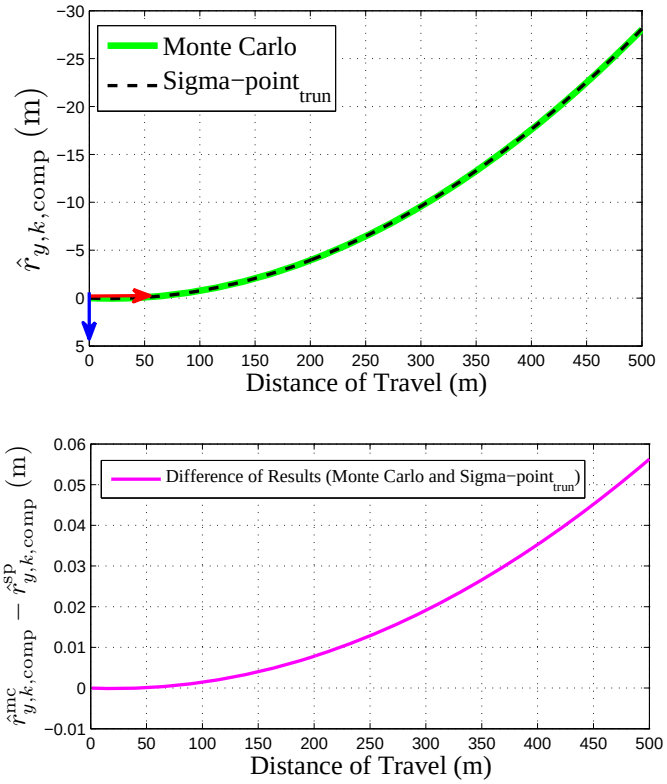

Figure 5.2: The compound translation of the camera in the $y-$axis direction versus distance of travel for the sigma-point method (using the truncated Gaussian) and brute-force Monte Carlo simulations (top panel) and the difference between them (bottom panel). These plots describe average VO performance and no bias correction has been performed on the paths.

point method (using ground truth knowledge). Figure 5.2 indicates that our proposed method based on sigma-points with the truncated Gaussian performs well against the brute-force Monte Carlo approach, but for only a fraction of the computational cost (i.e., tens of samples instead of tens of thousands of samples). Note that the frame-to-frame bias is compounded for 500 m of travel to generate the resulting camera paths described in Figure 5.2. No bias correction has been performed on these paths.

5.2.2 Effect of the Bias Correction Algorithm on the Simulation Results

In this section, simulation results for two cases are demonstrated. The first case describes an asymmetric configuration of landmarks (in image space) and the error terms corresponding to the estimated camera pose before and after bias correction are

illustrated and compared. In addition, the effect of changing parameters including the position of landmarks from the camera, the measurement noise standard deviation, and the disparity threshold on these error terms is shown. The second case covers a scenario that reflects more closely the real-world VO problems since it describes a structure in which the camera is tilted from horizontal and observes landmarks that are randomly located on the ground. Finally, the result of our bias correction algorithm using different methods (the sigma-point method (two versions) and Box's method (two versions)) is indicated for a 100-metre distance of travel.

Simulations for a Simplified Asymmetric Configuration

In this section, we show the outcome of our estimator before and after applying the bias correction algorithm. For the following simulations we utilize the asymmetric structure of the setup that is depicted in Figure 4.2. The simulations are performed for 70 time steps and illustrates that a 70-metre path is traversed by the robot. The estimated pose of the camera attached to the robot is then computed. At time step k, the compounded estimate, $\hat{\mathbf{T}}^{(k)}$, is obtained by multiplying all the estimated transformation matrices up to and including the one corresponding to the time step k ($k = 1 \dots K$) as follows:

$$\hat{\mathbf{T}}^{(k)} \quad = \quad \hat{\mathbf{T}}_k \times \ldots \times \hat{\mathbf{T}}_1 \quad k = 1 \dots K. \tag{5.26}$$

To show the simulation results, we first compute the error terms. To do this, we require to achieve the compounded true transformation matrix $\mathbf{T}_{\mathrm{t}}^{k}$ for $k = 1 \dots K$ as follows:

$$\mathbf{T}_{\mathrm{t}}^{(k)} \quad = \quad \mathbf{T}_{k,\mathrm{t}} \times \ldots \times \mathbf{T}_{1,\mathrm{t}} \quad k = 1 \dots K. \tag{5.27}$$

Before applying our bias correction algorithm, the difference between the estimated transformation matrix, $\hat{\mathbf{T}}^{(k)}$, and our ground truth, $\mathbf{T}_{\mathrm{t}}^{(k)}$, is denoted by $\mathbf{T}_{\mathrm{error}}^{(k)}$ and computed from the following expression

$$\mathbf{T}_{\mathrm{error}}^{(k)} = \hat{\mathbf{T}}^{(k)}(\mathbf{T}_{\mathrm{t}}^{(k)})^{-1} \tag{5.28}$$

After applying our bias correction algorithm, the compounded modified estimates are given by

$$\hat{\mathbf{T}}_{\mathrm{mod}}^{*,(k)} = \hat{\mathbf{T}}_{k,\mathrm{mod}}^{*} \times \ldots \times \hat{\mathbf{T}}_{1,\mathrm{mod}}^{*} \quad k = 1 \ldots K, \tag{5.29}$$

where $\hat{\mathbf{T}}_{k,\mathrm{mod}}^{*}$ are introduced in Section 5.1.2. The discrepancy between our modified estimate, $\hat{\mathbf{T}}_{\mathrm{mod}}^{*,(k)}$, and the true value, $\mathbf{T}_{\mathrm{t}}^{(k)}$, is called $\widetilde{\mathbf{T}}_{\mathrm{error}}^{(k)}$ which is given by

$$\widetilde{\mathbf{T}}_{\mathrm{error}}^{(k)} = \hat{\mathbf{T}}_{\mathrm{mod}}^{*,(k)}\,(\mathbf{T}_{\mathrm{t}}^{(k)})^{-1}. \tag{5.30}$$

Now we convert the transformation matrices $\mathbf{T}_{\mathrm{error}}^{(k)}$ and $\widetilde{\mathbf{T}}_{\mathrm{error}}^{(k)}$ to the camera pose estimate vectors $\boldsymbol{\pi}_{\mathrm{error}}^{(k)}$ and $\widetilde{\boldsymbol{\pi}}_{\mathrm{error}}^{(k)}$ using expressions (2.9) to (2.15). Then we decompose $\boldsymbol{\pi}_{\mathrm{error}}^{(k)}$ to the translation and orientation components associated with the $x-$, $y-$, and $z-$ axes as follows:

$$\boldsymbol{\pi}_{\mathrm{error}}^{(k)} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}_{\mathrm{error}}^{(k)}, \quad \boldsymbol{\rho}_{\mathrm{error}} = \begin{bmatrix} \rho_x \\ \rho_y \\ \rho_z \end{bmatrix}_{\mathrm{error}}^{(k)}, \quad \boldsymbol{\phi}_{\mathrm{error}} = \begin{bmatrix} \phi_x \\ \phi_y \\ \phi_z \end{bmatrix}_{\mathrm{error}}^{(k)}. \tag{5.31}$$

Similarly, $\widetilde{\boldsymbol{\pi}}_{\mathrm{error}}^{k}$ can be decomposed to its components. To better observe the improvement of the results after applying the bias correction algorithm we will superimpose the error terms before and after correction for bias on a single plot. A simulation

is performed to provide error terms in two cases when we vary the position of the landmarks with respect to the camera. We do this by moving the configuration of landmarks in a way that the position of the closest landmark is set to $(0, 0, z)$ ($z \in \{13\,\text{m}, 15\,\text{m}\}$) and the standard deviation of noise is fixed at 0.25 pixels (Figure 5.3). Another simulation is based on varying measurement noise standard deviation from 0.1 pixels to 0.25 pixels when the closest landmark is fixed at $(0, 0, 15\,\text{m})$ (Figure 5.4). The last simulation is performed to demonstrate the effect of varying the disparity threshold on error terms. The value of the disparity threshold is altered from 3 pixels to 4 pixels while other parameters are fixed (Figure 5.5). The simulations are conducted for 700 runs of experiments that only differ in their measurements (different realizations of noise are added to the same true state). Figures 5.3 to 5.5 show the average and standard deviation of the translation and orientation error terms corresponding to these 700 runs.

In Figure 5.3, the solid curves show the mean of 700 error terms and the dotted curves indicate the standard deviation. Two cases are compared and the results are shown above. The black curve shows the error terms when the closest landmark to the camera is located 13 metres away on the $z-$axis and the configuration of other four landmarks could be found in Figure 4.2. The blue curve shows the corresponding error terms after bias correction. The red curve shows the error terms when the landmarks are moved farther from the camera in the way that the closest landmark is located at 15 metres away on the $z-$axis. The magenta curve shows the corresponding error terms after bias correction.

In Figure 5.4, the solid curves show the mean of 700 error terms and the dotted curves indicate the standard deviation. Two cases are compared and the results are shown above. The black curve shows the error terms when the standard deviation of noise is 0.1 pixels and the closest landmark to the camera is located 15 metres away

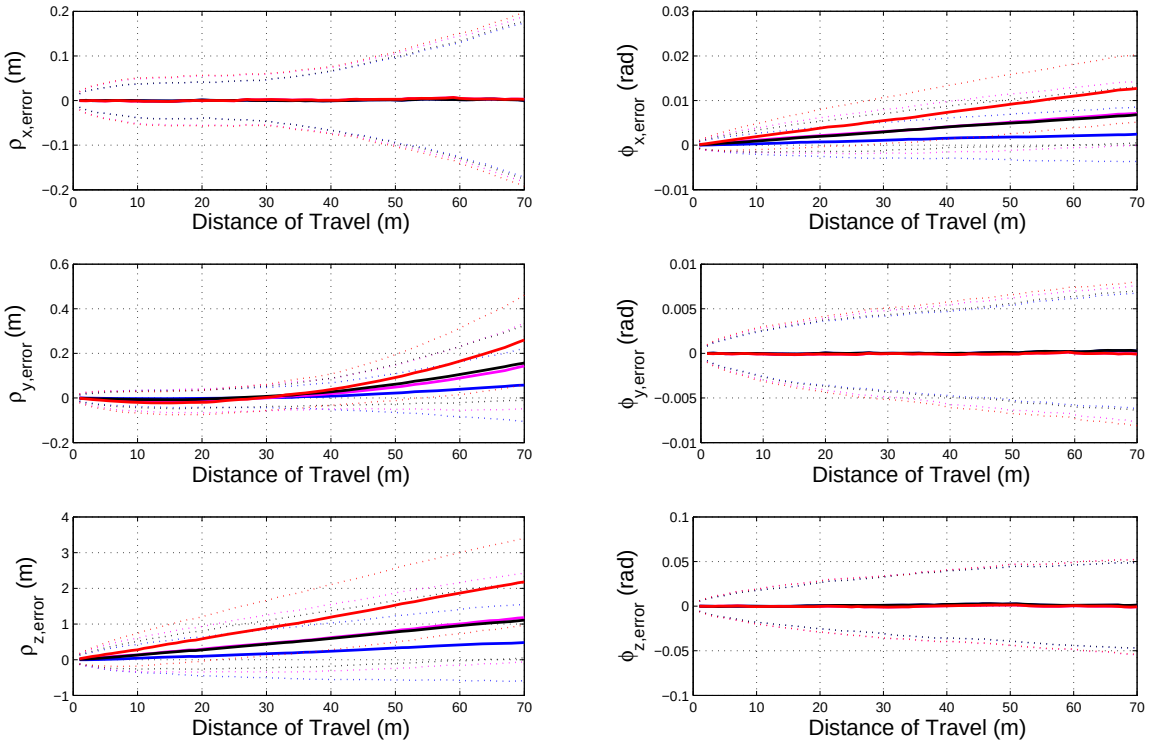

Figure 5.3: Left: Translation error terms in the direction of $x-$ (top), $y-$ (middle), and $z-$ (bottom) axes; right: orientation error terms in the direction of $x-$ (top), $y-$ (middle), and $z-$ (bottom) axes. The standard deviation of noisy measurements is 0.25 pixels and the disparity threshold is 4 pixels in the simulations. The solid curves show the mean of 700 error terms and the dotted curves indicate the standard deviation of these terms.

on the $z-$axis and the configuration of other four landmarks could be found in Figure 4.2. The blue curve shows the corresponding error terms after bias correction. The red curve shows the error when the configuration of landmarks is fixed and we increase the standard deviation to 0.25 pixels. The magenta curve shows the corresponding error after bias correction.

In Figure 5.5, the solid curves show the mean of 700 error terms and the dotted curves indicate the standard deviation. Two cases are compared and the results are shown above. The black curve shows the error terms when the disparity threshold is 3 pixels, the standard deviation is 0.25 pixels, and the closest landmark to the camera

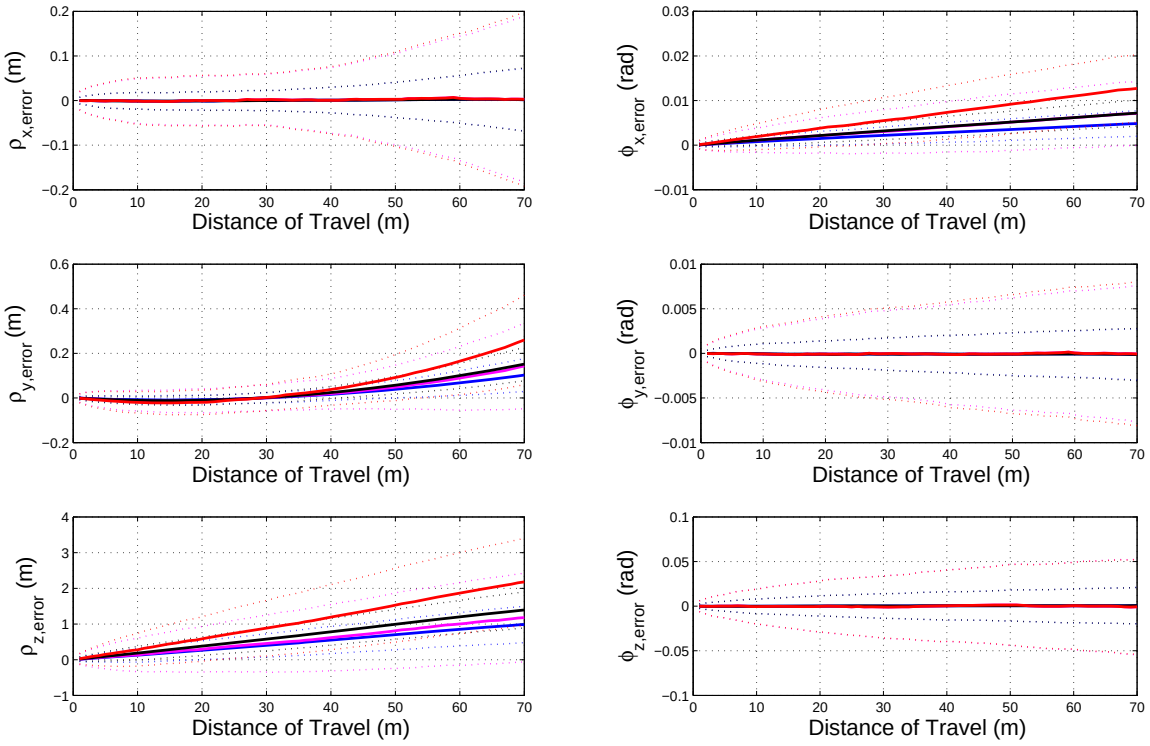

Figure 5.4: Left: Translation error terms in the direction of $x-$ (top), $y-$ (middle), and $z-$ (bottom) axes; right: orientation error terms in the direction of $x-$ (top), $y-$ (middle), and $z-$ (bottom) axes. The solid curves show the mean of 700 error terms and the dotted curves indicate the standard deviation of these terms.

is located 15 (bottom) metres away on the $z-$axis and the configuration of other four landmarks could be found in Figure 4.2. The blue curve shows the corresponding error terms after bias correction. The red curve shows the error terms when the configuration of landmarks and the measurement standard deviation are fixed and we only increase the value of the disparity threshold to 4 pixels. The magenta curve shows the corresponding error terms after bias correction.

Based on these Figures, we observe bias in the orientation of camera about the $x-$axis while there is no bias in the translation of the camera in the $x-$axis. For instance, in Figure 5.4 bias observed in the orientation of camera after it travels 70 metres is 0.01 radians when the noise standard deviation is 0.25 pixels. In addition,

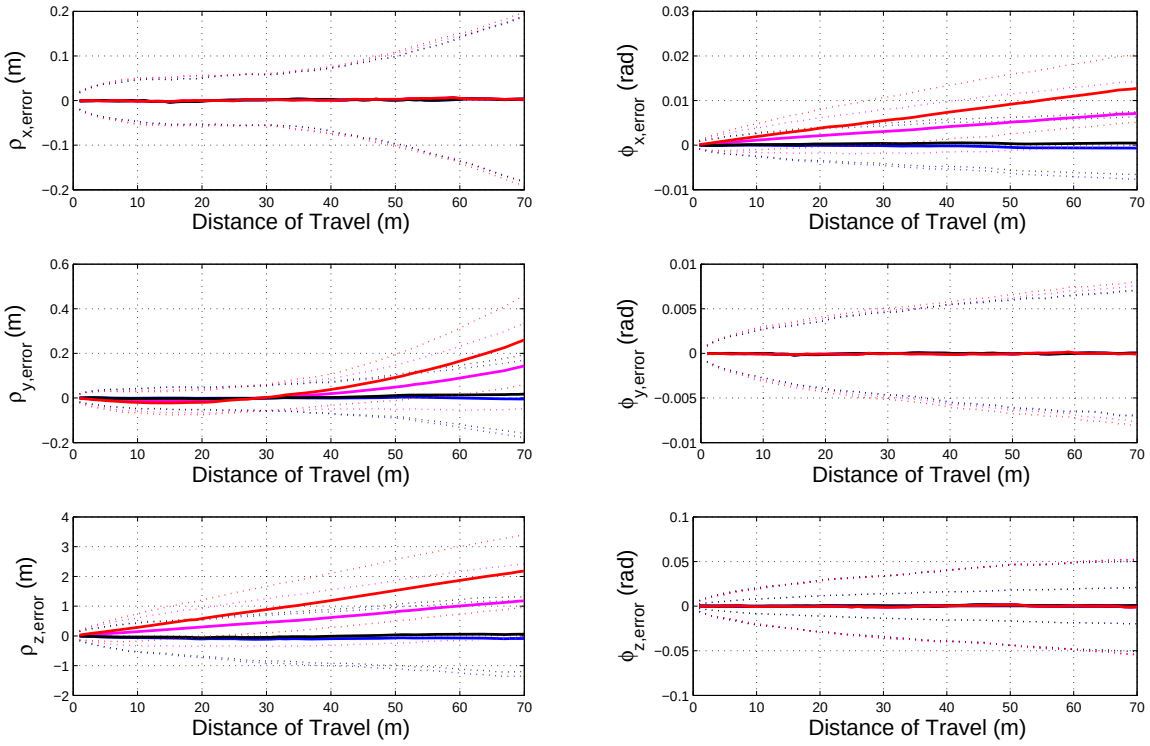

Figure 5.5: Left: Translation error terms in the direction of $x-$ (top), $y-$ (middle), and $z-$ (bottom) axes; right: orientation error terms in the direction of $x-$ (top), $y-$ (middle), and $z-$ (bottom) axes. The solid curves show the mean of 700 error terms and the dotted curves indicate the standard deviation of these terms.

these figures show bias in the translation of camera in the direction of the $y-$axis while no bias is observed in the orientation about $y-$axis. Figure 5.4 shows 0.26 metres translation bias for 70-metre distance of travel and 0.25 pixels noise standard deviation. In all of these figures, bias is observed in the translation in the $z-$axis direction but no bias is observed in the orientation of camera about $z-$axis. For example, Figure 5.4 shows 2.18 metres translation bias when the camera travels 70 metres and the noise standard deviation is 0.25 pixels.

Overall, when bias is observed in either translation or orientation of the camera, it is increased when landmarks are located farther with respect to the camera (refer to Figure 5.3), the standard deviation of the measurement noise is larger (refer to Figure

5.4), or the disparity threshold has a greater value (refer to Figure 5.5). Furthermore, the modified estimates lead to the smaller error terms compared to the ones before applying bias correction.

Simulations for a Case Similar to the Real-world VO problem

In another set of simulations, in order to demonstrate the effectiveness of our bias correction method, we investigate a scenario that represents the real-world VO problem. In this analysis, a robot moves forward along a straight line with a stereo camera attached to it but tilted down 15^o from the horizontal, and observes a set of randomly located landmarks on the ground. The distribution of landmarks in our simulations is shown in Figure 5.6 for an area of $150\,\mathrm{m} \times 40\,\mathrm{m}$. The area is expanded in the $z-$axis direction when the robot travels a longer distance.

In the frame-to-frame motion estimation process, the camera captures a number of landmarks in its field of view; it moves one metre in the next frame and captures landmarks that are in its new field of view. The set of common landmarks captured in these two sequential frames are marked. They are used in our VO estimation algorithm for estimating motion (see Figure 5.7).

In the process of investigating bias, we first illustrate the identified bias using the average of 700 camera pose estimates in a distance of 500 metres in Figure 5.8. The mean of path estimates deviates about five metres downward from ground truth for $\sigma = 0.25$ pixels and becomes larger for a noisier setup.

Further analysis is performed to study the behaviour of bias when the angle the camera is tilted from horizontal (z-axis in Figure 5.9) and disparity threshold (Figure 5.10) are changed. The result demonstrates that bias is larger for a smaller camera tilt from horizontal. When the camera is tilted with a smaller angle from horizontal, there are more far observed landmarks in its field of view. Consequently, an increase

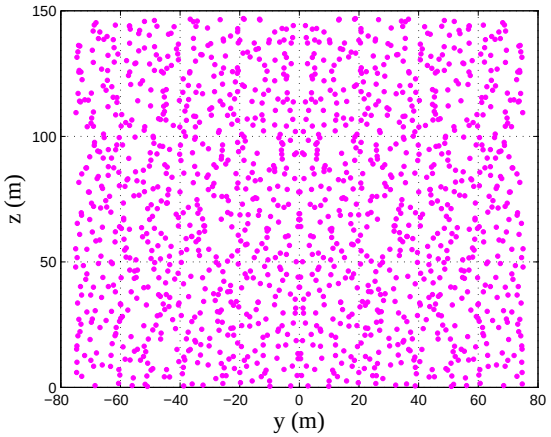

Figure 5.6: Configuration of landmarks that are randomly located on the ground.

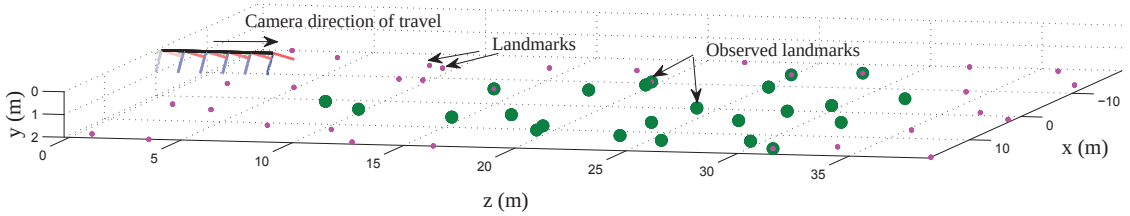

Figure 5.7: A stereo camera is tilted 15^o down from horizontal and moves above the ground capturing landmarks that are in its field of view. The camera views a set of landmarks, then it translates to the next frame and captures landmarks that it observes. At each time step, the common landmarks of the two sequential frames are found and referred to as observed landmarks. The observed landmarks in five time steps are indicated in green and the landmarks shown in magenta are not observed. In this picture, the stereo camera travels five metres without changing its orientation.

in the number of far landmarks observed by the camera will result in a larger bias. This is indicated in Figure 5.9 where the camera is tilted 15^o in one case and 25^o in another case. The results show an increase of 0.03 metres in bias when the camera angle varies from 25^o to 15^o.

Figure 5.10 shows that the bias is larger for a bigger disparity threshold. Once we increase the disparity threshold, we truncate the measurement distribution more. Therefore, the nonlinearity of the system is increased and it causes a larger bias in the result. To demonstrate this, we performed simulations with disparity threshold

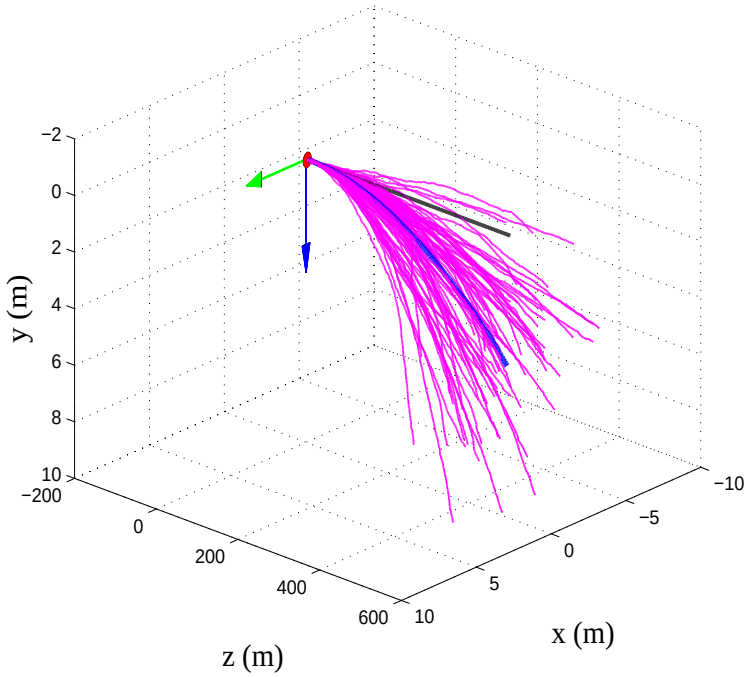

Figure 5.8: The camera travels 500 metres forward (refer to Figure 5.7). The ground-truth path is indicated in black. The travelled path of the camera is estimated and indicated in magenta and the mean of these runs are shown in blue. The mean of the estimates deviates downward from our ground truth.

of 3 pixels and 4 pixels and the output of the estimation algorithm in each case is shown. The results show that when the disparity threshold increases from 3 pixels to 4 pixels, bias is increased by 0.15 metres after 100 metres of travel.

In Figure 5.11 different methods of bias correction are implemented to correct for the biased VO estimates. The camera is 15^o tilted down with respect to the horizontal and travels 100 metres (see Figure 5.7). It is observed from Figure 5.11 that the truncated Box method fails in correcting for bias since the method does not perform well when the disparity threshold discards measurements. However, the results demonstrate that our proposed method, the truncated sigma-point method, provides an improved estimate. The outputs of two versions of the truncated sigma-point method are considered in this simulation. The first version is the truncated

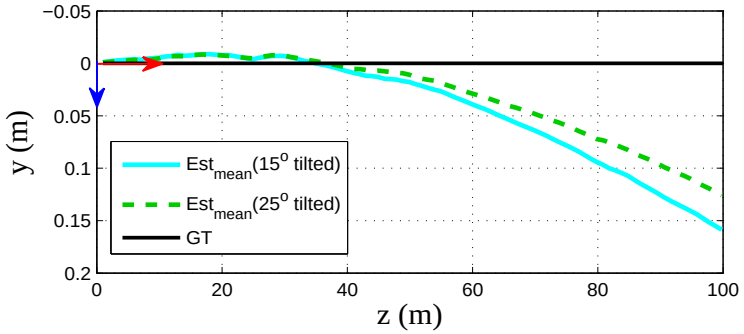

Figure 5.9: The camera is tilted with two different angles (15^o and 25^o) and the mean of estimates are illustrated above. It is observed that bias is larger when the camera is tilted with smaller angle from horizontal. y- and z- axes of the initial frame with respect to which the camera is tilted is shown. No bias correction has been performed on the paths.

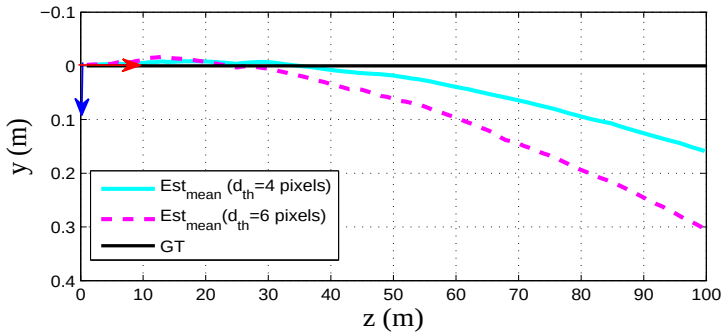

Figure 5.10: The disparity threshold (d_{th}) is set to 4 pixels and 6 pixels in two cases and the mean of estimates is illustrated for each case. The result shows a larger downward bias when d_{th} is larger. y- and z- axes of the initial frame with respect to which the camera is tilted and its estimated path is computed is depicted above. No bias correction has been performed on the paths.

sigma-point method based on ground truth, indicated by the subscript 'trun-GT' (bias is estimated using (5.9) and the biased estimates are corrected for bias using (5.11)). The mean of the corrected estimates is close to ground truth with a good approximation.

The second version is the truncated sigma-point method based on the bootstrap approach, indicated by the subscript 'trun' (bias estimation is performed using (5.23) and the estimates are corrected for bias using (5.24)). Figure 5.11 shows a 95% bias

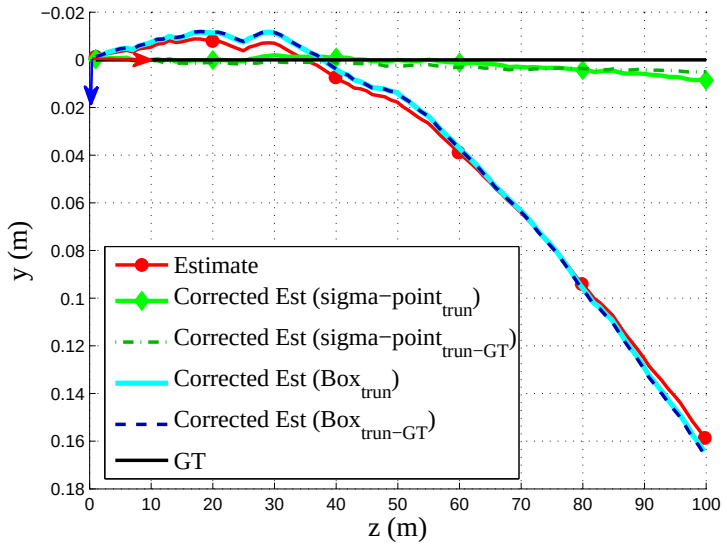

Figure 5.11: The mean of the estimated paths for a camera that travels 100 metres in the z-axis direction (refer to Figure 5.7) is shown where different methods of bias correction are applied. The truncated sigma-point bias correction technique has successfully removed bias from the estimates. The results associated with the truncated Box method describe that this method fails due to the existence of the disparity threshold.

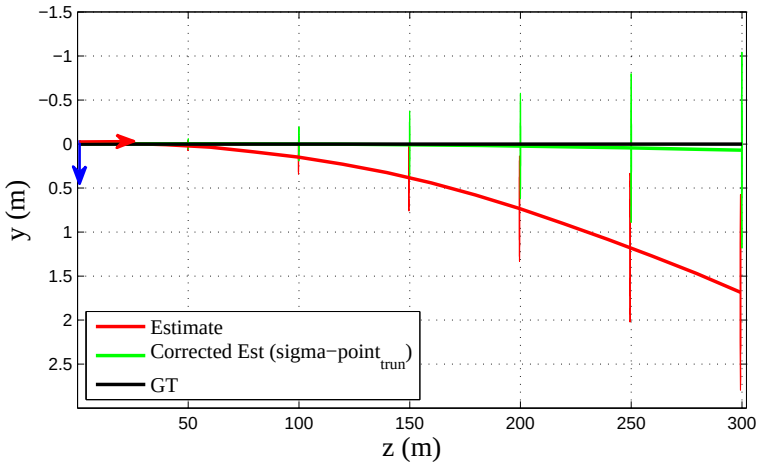

Figure 5.12: The mean curves of the estimated paths for a camera that travels 300 metres in the z-axis direction (refer to Figure 5.7) are shown before and after bias correction. The truncated sigma-point bias correction technique has successfully removed bias from the estimates. The uncertainty bars associated with each curve are also indicated above.

reduction result after implementing this bias correction method. The average of 700 path estimates are shown for a longer distance of travel in Figure 5.12; however, only the truncated sigma-point method based on the bootstrap is used to correct for bias. Additionally, the uncertainty bars are indicated to demonstrate the spread of the estimates. It is observed that bias is greatly reduced in the average of the estimates after applying the bias correction algorithm.

It was pointed out previously that in real-world VO problems we only have access to one set of measurements at each time step. Therefore, we are not able to compute the average of a number of modified estimates to observe bias correction. We conduct a simulation to show how our bias correction performs when it is applied to the biased estimated path of the camera in a distance of 300 metres. The simulation is performed for 700 experiments (only two experiments are shown in Figure 5.13) that only differ in the realizations of the noise added to produce input measurements for the estimation algorithm. According to the results indicated in Figure 5.13, it is observed that the bias correction based on the truncated sigma-point approach successfully removes a large amount of bias from the estimates in a distance of 300 metres. Additionally, we computed the number of improved estimates at each time step to demonstrate that the percentage of estimates on which our correction algorithm provides improvement increases as the camera travels a longer distance (refer to Figure 5.14). Based on this result, we can predict that for a sufficiently long distance of travel, all the modified estimates to which our bias correction algorithm is applied will eventually have some improvements over the original estimates.

The truncated sigma-point method estimates bias with high accuracy and tracks the results obtained by brute-force Monte Carlo simulations. (refer to (5.2)). The benefit of the sigma-point method is that it greatly reduces the computational cost of applying our bias estimation algorithm. While the Monte Carlo simulations require a

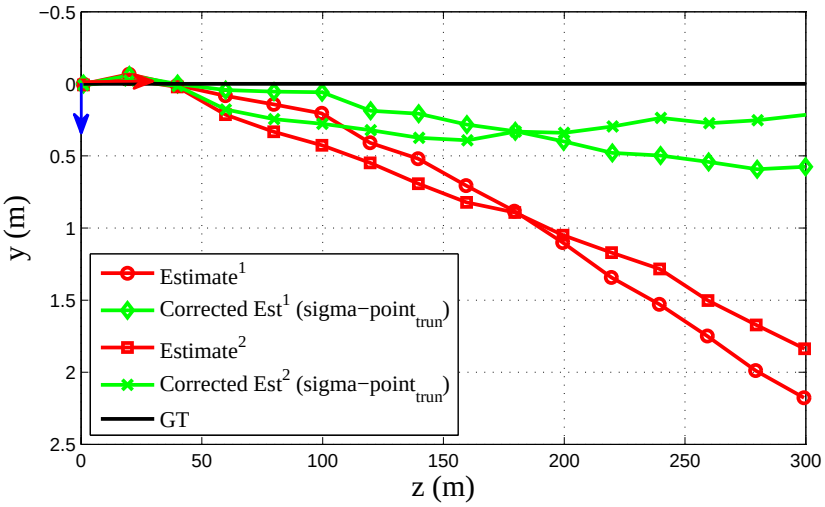

Figure 5.13: The estimated path of the camera that travels 300 metres in the direction of $z-$axis is shown for two independent experiments (refer to Figure 5.7 to see the configuration of landmarks and the camera). Different realizations of noise are added to generate the input measurements of each experiment. The truncated sigma-point method is used for bias correction along the path.

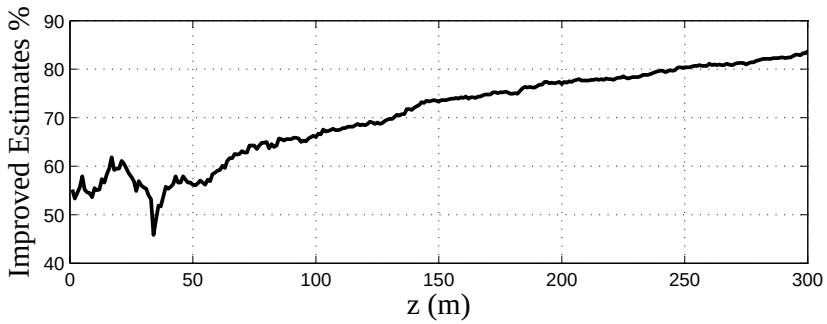

Figure 5.14: A simulation is performed for 700 experiments of pose estimation, two of which are shown in Figure 5.13, and the number of improved experiments (improved estimates) is indicated as a percentage of total experiments.

large number of noisy samples, the sigma-point method only requires $2\,N+1$ selected sigma-points where $N \propto J$ and J is the number of observed landmarks in each time step. Although this method is not still fast enough for online implementation, it is a good starting point for estimating bias using only stereo images. The sigma-point method could be easily parallelized and run on a GPU since each sample simply runs

its own copy of the overall estimator.

In this section, we simulated a scenario in which the camera's angle from horizontal remains constant as the robot moves forward. We observed bias in the camera pose estimate even though the orientation of the camera did not change. We determined by simulations that the estimate was biased due to the asymmetric configuration of landmarks with respect to the camera. The purpose of this section was to show bias in the estimate and propose methods to correct it. The scenario we described in Figure 5.6 was a simple example similar to the real-world VO problems and it showed bias in the camera pose estimate. We observed that the bias correction algorithms worked well for this scenario. More complicated scenarios could be modelled that resemble the real-world VO problems better by involving the camera orientation changes as the robot travels. Changes in the camera orientation including left or right turns and also upwards or downwards tilt of the camera from the horizontal, could be examined. For example, in a scenario that a robot climbs a hill, we should first model the gradient of the hill. Then, the pose change of the camera when the robot climbs the hill is considered as ground truth and the bias correction algorithms can be applied. We did not provide these examples since modelling of ground truth for different scenarios and running bias correction algorithms for them requires extensive computations and is beyond the scope of this thesis. We believe that the examples provided were sufficient to demonstrate that we achieved the objective of the thesis which is identifying bias and removing it from the estimates using the bias correction algorithms.

Chapter 6

Summary

6.1 Thesis Contributions

In this thesis, we first demonstrated the existence of bias in the visual odometry problem using brute-force Monte Carlo simulations. We then discussed three techniques for estimating bias in VO: Monte Carlo sampling, the sigma-point method, and Box's [14] analytical technique. A number of challenges were revealed during this study. Firstly, we identified that applying a disparity threshold to measurements in stereo problems causes further complications to bias estimation, but that this can be overcome by adjusting the Gaussian measurements statistics in a pre-processing step to account for the disparity threshold truncation process. We compared the various bias estimation methods on a one-landmark stereo triangulation problem as well as a frame-to-frame VO problem. Secondly, eliminating the assumption of knowing ground truth in our analysis was not straightforward since we intended to estimate bias while relying on the original estimate that was itself biased. We employed the idea of the bootstrap [19, 50] in statistics to handle this challenge.

Finally, we combined all these ingredients to propose a novel methodology referred

to as the truncated sigma-point bias correction algorithm that uses only stereo camera images of landmarks to estimate and correct bias. This approach is highly flexible in that it considers the position estimation algorithm as a black box. Any other position estimation algorithm could be easily used along with our bias correction approach. Therefore, the effect of any extra nonlinearity added to the position estimation setup will be reflected and included in this bias correction algorithm. We demonstrated that this novel method for bias correction effectively reduced 95% of bias based on simulation results for a scenario that closely approximate a real-world VO problem.

6.2 Future Work

The main intent of this study was to estimate bias by evaluating the sample estimates transformed by the nonlinear position estimation algorithm. To this end, we utilized biased sample estimates in our approach without attempting to extract bias and model it in terms of the system's parameters.

Our presented technique performed well compared to the brute-force Monte Carlo method, but using a small fraction of the computational cost. To be precise, while the Monte Carlo simulations require more than 10000 noisy samples (in some cases up to 40000 samples), the sigma-point method only requires $2\,L+1$ selected sigma-points where $L=8J$ is the length of the observation vector and J is the number of observed landmarks in each time step. In our simulations for a case similar to the real-world VO problem, $J \in \{13, \cdots, 25\}$. Thus, the bundle adjustment algorithm, the most time-consuming part of our state estimation problem, runs $2\,L+1 \in \{209, \cdots, 401\}$ times. Based on these numbers, our method runs 25 to 48 times faster than using Monte-Carlo method. Still, our method may be too slow for online implementations, but serves as a good proof-of-concept starting point. Future research could put an

emphasis on speeding up computation by parallelization.

Another future work effort may be to conduct more simulations to assess various scenarios that approximate the real-world problems. In this contribution, bias was estimated and corrected in the camera pose estimates for the cases in which the robot travels from one point to another one without changing its orientation.

Lastly, the proposed algorithm should be applied on real data sets to experimentally determine what improvement this offers on real stereo image sequences. Real data sets are in the format of stereo images and should be processed based on the steps in the VO pipeline described in Chapter 2. Such data will be different from the idealized simulation data analyzed in this thesis. Pre-processing adapting the data is needed to obtain measurements in a form suitable for motion estimation.

Bibliography

[1] Y. Amihud, C. M. Hurvich, "Predictive Regressions: A Reduced-Bias Estimation Method," *Statistics Working Papers Series*, 2004.

[2] `http://asrl.utias.utoronto.ca/~tdb/`.

[3] H. Asada and M. Brady, "The curvature primal sketch," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1), pp. 2–14, 1986.

[4] N. Ayache and C. Hansen, "Rectification of Images for Binocular and Trinocular Stereovision," *INRIA Technical Report 860*, 1988.

[5] T. Barfoot, J. R. Forbes, and P. T. Furgale, "Pose estimation using linearized rotations and quaternion algebra," *Acta Astronautica*, 2010.

[6] T. Barfoot, "'Bells and Whistles' for Bundle Adjustment," University of Toronto Institute for Aerospace Studies (UTIAS), ASRL Lab, 2012.

[7] H. Bay, A. Ess, T. Tuytelaars,and L. V. Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, 110(3), pp. 346–359.

[8] S. Baker, S. K. Nayar, and H. Murase, "Parametric feature detection," *International Journal of Computer Vision*, 27(1), pp. 27–50, 1998.

[9] M. J. Box, "Bias in nonlinear estimation," *Journal of the Royal Statistical Society.* Series B (Methodological), 33(2), pp. 171–201, 1971.

[10] G. Babbar, P. Bajaj, A., Chawla, and M. Gogna,"Comparative Study Of Image Matching Algorithms," *International Journal of Information Technology and Knowledge Management*, Volume 2, No. 2, pp. 337–339, 2010.

[11] C. Belta and V. Kumar, "Euclidean Metrics for Motion Generation on SE(3)," in *Proc.of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 216(1), pp. 47–60, 2002.

[12] A. Björck, "Numerical methods for least squares problems," SIAM Publications, Philadelphia, PA, 1996.

[13] S. C. Botelho, P. Drews, G. L. Oliveira, and M. S. Figueiredo, "Visual odometry and mapping for underwater autonomous vehicles," in *Robotics Symposium (LARS)*, 2009 6th Latin American, pp. 1–6, 2009.

[14] M. J. Box, "Bias in nonlinear estimation," Journal of the Royal Statistical Society. Series B (Methodological), 33(2), pp. 171–201, 1971.

[15] D. C. Brown, "A Solution to the General Problem of Multiple Station Analytical Stereotriangulation," RCA-MTP data reduction Technical Report no. 43 (or AFMTC 58-8), Patrick Airforce Base, Florida, 1958.

[16] D. C. Brown, "The bundle adjustment-progress and prospects," *Int. Archives Photogrammetry*, 21(3), 1976.

[17] M. Z. Brown, D, Burschka, and G. D. Hager, "Advances in computational stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8), pp. 993–1008, 2003.

[18] Y. Cheng, M. W. Maimone, and L. Matthies, "Visual odometry on the mars exploration rovers," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 54–62, 2006.

[19] M. R. Chernick, "Bootstrap methods: a guide for practitionars and researchers," Hoboken, N.J. : Wiley-Interscience, 2008.

[20] R. Cohn and J. Russell, "Selection bias," VSD Publishers, 2012.

[21] R. A. Coogle, L. D. Smith, and W. D. Blair, "Mitigating the Bias in Converted Bistatic Radar Measurements Using the Unscented Transform," *IEEE Aerospace Conf.*, Big Sky, Montana, 2013.

[22] J. L. Crassidis and F. L. Markley, "Unscented Filtering for Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 4, pp. 536–542, 2003.

[23] B. Cyganek and J. P. Siebert, "An Introduction to 3D Computer Vision Techniques and Algorithms," Chichester, U.K: J. Wiley & Sons, 2009.

[24] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1052–106, 2007.

[25] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, "A Modern Introduction to Probability and Statistics," Springer Texts in Statistics, pp. 285–297, 2005.

[26] G. M. T. DEleuterio, "Dynamics of an Elastic Multibody Chain: Part C Recursive Dynamics," Dynamics and Stability of Systems, 7(2), pp. 61–89, 1992.

[27] R. Deriche,. Using Cannys criteria to derive a recursively implemented optimal edge detector, "International Journal of Computer Vision", 1(2), pp. 167–187.

[28] P. Diaconis and B. Efron, "Computer-intensive methods in statistics," Sci. Am. 248 , pp. 116–130, 1983.

[29] G. Dubbleman and F.C.A. Groen, "Bias Reduction for Stereo Based Motion Estimation with Applications to Large Scale Visual Odometry," in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pp. 2222–2229, 2009.

[30] G. Dubbelman, P. Hansen, and B. Browning, "Bias Compensation in Visual Odometry," *IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2828–2835, 2012.

[31] B. Efron, "Bootstrap methods; another look at the jackknife," Ann. Statist. 7, pp. 1–26, 1979.

[32] B. Efron, "The Jackknife, the Bootstrap, and Other Resampling Plans," SIAM, Philadelphia, 1982.

[33] N. El-Sheimy, E. H. Shin, and X. Niu, "Kalman filter face-off: extended vs. Unscented kalman filters for integrated GPS and MEMS inertial," in *Proceedings of the International Symposium on Global Navigation Satellite Systems (GNSS '06)*, pp. 48–54, 2006.

[34] B. Efron and G. Gong, "A leisurely look at the bootstrap, the jackknife and cross-validation," Am. Statist. 37, pp. 36–48, 1983.

[35] B. Efron and R. Tibshirani, "Bootstrap methods for standard errors: Confidence intervals and other measures of statistical accuracy," Statist. Sci. 1 , pp. 54–77, 1986.

[36] O. Faugeras, "Three-Dimensional Computer Vision," A Geometric Viewpoint , MIT Press, 1993.

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

[38] R. Fletcher, Roger, "Practical methods of optimization," (2nd ed.). New York: John Wiley & Sons, 1987.

[39] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part II– Matching, robustness, optimization, and applications," *IEEE Robotics and Automation Magazine*, Vol. 19, issue 2, 2012.

[40] P. Fua, "A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features," *INRIA Technical Report 1369*, 1991. .

[41] P. T. Furgale, P. Carle, J. Enright, and T. D. Barfoot, "The Devon Island rover navigation dataset," *Intern. Journal of Robotics Research*, 2010.

[42] C. F. Gauss, "Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections (Translated by C. H. Davis, 1963)," New York: Dover, 1809.

[43] S. Granshaw, "Bundle adjustment methods in engineering photogrammetry," *Photogrammetric Record*, 10(56), pp. 181–207, 1980.

[44] S. Gratton, A. S. Lawless, and N. K. Nichols, "Approximate Gauss-Newton methods for nonlinear least squares problems." SIAM Journal on Optimization, 18(1), pp. 106-132, 2007.

[45] L. L. Grewe, A. C. and Kak, "Stereo vision," in *Handbook of Pattern Recognition and Image Processing: Computer Vision*, Vol. 2 (ed. T.Y. Young), Academic Press, 1994.

[46] C. Harris, and M. Stephens," A combined corner and edge detection," *In Proceedings of The Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[47] R.I. Hartley, and A. Zisserman, "Multiple View Geometry in Computer Vision," (2nd ed), Cambridge University Press, 2004.

[48] D. Helmick, Y. Cheng, S. Roumeliotis, D. Clouse, and L. Matthies, "Path following using visual odometry for a Mars rover in highslip environments," in *IEEE Aerospace Conference*, Big Sky, Montana, USA, 2004.

[49] P. C. Hughes, "Spacecraft Attitude Dynamics," John Wiley & Sons, New York, 1986.

[50] T. Hesterberg, "Bootstrap," *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(6), pp. 497–526, 2011.

[51] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3946–3952, 2008.

[52] M. Irani, and P. Anandan, "About Direct Methods," *International Workshop on Vision Algorithms*, Corfu, Greece, 1999.

[53] K. Ito and K. Xiong. "Gaussian filters for nonlinear filtering problems," *IEEE Transactions on Automatic Control*, 45(5), pp. 910–927, May 2000.

[54] S. Julier and J. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Technical Report, Dept. of Engineering Science, University of Oxford, Oxford, 1996.

[55] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, 45, pp. 477-482, 2000.

[56] L. Kitchen, and A. Rosenfeld, "Gray-level corner detection," *Pattern Recognition Letters*, 1(2), pp. 95–102, 1982.

[57] M. Kline, "Mathematical Thought from Ancient to Modern Times," Vol(3), Oxford University Press Inc., New York, NY, USA, 1972.

[58] K. Konolige, M. Agrawal, and J. and Solà, "Large scale visual odometry for rough terrain," In *Proc. International Symposium on Research in Robotics (ISRR)*, 2007.

[59] S. P. Kothari and J. Shanken, "Book-to-Market, Dividend Yield, and Expected Market Returns: A Time-Series Analysis," *Journal of Financial Economics*, 18, pp. 169–203, 1997.

[60] A. Lambert, P. T. Furgale, T. D. Barfoot, and J. Enright, "Visual Odometry Aided by a Sun Sensor and Inclinometer," in *Proc. of IEEE Aerospace Conference*, pp. 1–12, 2011.

[61] A. Lambert, P. T. Furgale, T. D. Barfoot, and J. Enright, "Field Testing of Visual Odometry Aided by a Sun Sensor and Inclinometer," *Journal of Field Robotics*, 29(3), pp. 426–444, May-June 2012.

[62] A. Lambert, "Visual odometry aided by a sun sensor and inclinometer," *Master's thesis*, Inst. for Aerospace Studies, University of Toronto, 2011.

[63] T. Lefebvre, H. Bruyninckx, and J. De Schutter, "Kalman filters for non-linear systems: a comparison of performance," *International Journal of Control*, 77(7), pp. 639–653, 2004.

[64] E. L. Lehmann and G. Casella, "Theory of Point Estimation," Springer, 1998.

[65] K. Levenberg, "A Method for the Solution of Certain Problems in Least-Squares," *Quarterly Applied Math.* 2, pp. 164–168, 1944.

[66] D. Lowe, "Distinctive image features from scale-invarient keypoints," International Jornal of Computer Vision, 60(2), pp. 91-110, 2004.

[67] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars exploration rovers," *Journal of Field Robotics*, 24(3), pp. 169–186, 2007.

[68] D. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *SIAM Journal Applied Math*, Vol(11), pp. 431–441, 1963.

[69] J. Marshall, T. D. Barfoot, and J. Larsson, "Autonomous underground tramming for center-articulated vehicles," *Journal of Field Robotics*, 25(6-7), pp. 400–421, 2008.

[70] L. Matthies, S. A. Shafer, "Error modeling in stereo navigation," Computer Science Dept., Carnegie Mellon University, Paper 1593, 1986.

[71] L. Matthies, "Dynamic stereo vision," *Ph.D. thesis*, Computer Science Dept., Carnegie Mellon University, 1989.

[72] C. McManus T. D. and Barfoot, "A Serial Approach to Handling High-Dimensional Measurements in the Sigma-Point Kalman Filter," I*n Proceedings of Robotics: Science and Systems (RSS)*, P29. Los Angeles, USA, 2011.

[73] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: A System for Large-Scale Mapping in Constant-Time using Stereo," *International Journal of Computer Vision*, pp. 1–17, 2010.

[74] R. M. Murray, Z. Li, and S. S. Sastry, "A matematical introduction to robotic manipulation," CRC Press, 1994.

[75] R. van der Merwe, "Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models," PhD thesis, Oregon Health & Science University, OGI School of Science & Engineering, 2004.

[76] R. van der Merwe, E. A. Wan, and S. I. Julier, "Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor Fusion: Applications to Integrated Navigation," AIAA Guidance, *Navigation and Contol Conference*, Providence, RI, 2004.

[77] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," *Ph.D. thesis*, Stanford University, 1980.

[78] J. J. Mor*é*, "The Levenberg-Marquardt Algorithm: Implementation and Theory," Numerical Analysis, ed. G. A. Watson, Lecture Notes in Mathematics 630, Springer Verlag, pp. 105–116, 1977.

[79] D. Nister, O. Naroditsky, and J. Bergen,"Visual Odometry," in *International Conference on Computer Vision and Pattern Recognition*, pp. 652–659, 2004.

[80] D. Nister, O. Naroditsky, and J., and Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, 23(1), pp. 3–20, 2006.

[81] J. Nocedal, and S. Wright, "Numerical optimization," New York: Springer, 1999.

[82] J.M. Ortega, and W.C. Rheinboldt, "Iterative Solution of Nonlinear Equations in Several Variables," Academic Press, New York, 1970.

[83] K. Paler, J. Foglein, J. Illingworth, J. and Kittler, "Local ordered greylevels as an aid to corner detection," *Pattern Recognition*, 17(5), pp. 535–543, 1984.

[84] D. V. Papadimitriou and T. J. Dennis, "Epipolar line estimation and rectification for stereo image pairs," *IEEE Transactions on Image Processing*, 5(4), pp. 672–676, 1996.

[85] J. Rehder, K. Gupta, S. T. Nuske, and S. Singh, "Global pose estimation with limited gps and long range visual odometry," *IEEE Conference on Robotics and Automation*, May. 2012.

[86] K. Rohr, "Towards model-based recognition of human movements in image sequences." *Computer Vision, Graphics, and Image Processing,* 59(1), 94–115, 1994.

[87] E. Rosten, and T. Drummond, "Fusing points and lines for high performance tracking," *In IEEE International Conference on Computer Vision*, volume 2, pp. 1508–1511, 2005.

[88] E. Rosten, and T. Drummond, "Machine learning for high-speed corner detection," *In European Conference on Computer Vision*, volume 1, pp. 430-443, 2006.

[89] G. Sibley, G. Sukhatme, and L. Matthies, "The iterated sigma point kalman filter with applications to long range stereo," *In Proceedings of Robotics Science and Systems*, Philadelphia, Pennsylvania, USA, 2006.

[90] R. F. Stambaugh, "Predictive Regressions," *NBER Technical Working Papers 0240*, National Bureau of Economic Research, Inc, 1999.

[91] R. Szeliski, "Computer Vision: Algorithms and Applications," London, New York, Springer, pp. 305, 2011.

[92] J. P. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz, "A new approach to vision-aided inertial navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4161–4168, 2010.

[93] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, "Bundle Adjustment – A Modern Synthesis," *International Workshop on Vision Algorithms* 1883 pp. 298–372, 2000.

[94] E. Trucco and A. Verri, "Introductory Techniques for 3-D Computer Vision", Prentice-Hall, 1998.

[95] Y. Wang, "GaussNewton method," Wiley Interdisciplinary Reviews: Computational Statistics, 4(4), pp. 415–420, 2012.

[96] P. H. Winston (Editor), B. Horn, M. Minsky, Y. Shirai, and D. Waltz, "The Psychology of Computer Vision," McGraw-Hill, New York, pp. 267–272, 1975.

[97] P. R. Wolf and C. D. Ghilani, "Adjustment Computations: Statistics and Least Squares in Surveying and GIS," John Wiley & Sons, 1997.

[98] L. Wu, J. Ma and J. Tian, "A self-adaptive unscented Kalman filtering for underwater gravity aided navigation," *Proc. IEEE Position Location Navig. Symp.*, pp. 142–145, 2010.