MONOCULAR VISION BASED LOCALIZATION AND MAPPING

by

MICHAL JAMA

M.S., Wroclaw University of Technology, Poland, 2007

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2011

Abstract

In this dissertation, two applications related to vision-based localization and mapping are considered: (1) improving navigation system based satellite location estimates by using on-board camera images, and (2) deriving position information from video stream and using it to aid an auto-pilot of an unmanned aerial vehicle (UAV).

In the first part of this dissertation, a method for analyzing a minimization process called bundle adjustment (BA) used in stereo imagery based 3D terrain reconstruction to refine estimates of camera poses (positions and orientations) is presented. In particular, imagery obtained with pushbroom cameras is of interest. This work proposes a method to identify cases in which BA does not work as intended, i.e., the cases in which the pose estimates returned by the BA are not more accurate than estimates provided by a satellite navigation systems due to the existence of degrees of freedom (DOF) in BA. Use of inaccurate pose estimates causes warping and scaling effects in the reconstructed terrain and prevents the terrain from being used in scientific analysis. Main contributions of this part of work include: 1) formulation of a method for detecting DOF in the BA; and 2) identifying that two camera geometries commonly used to obtain stereo imagery have DOF. Also, this part presents results demonstrating that avoidance of the DOF can give significant accuracy gains in aerial imagery.

The second part of this dissertation proposes a vision based system for UAV navigation. This is a monocular vision based simultaneous localization and mapping (SLAM) system, which measures the position and orientation of the camera and builds a map of the environment using a video-stream from a single camera. This is different from common SLAM solutions that use sensors that measure depth, like LIDAR, stereoscopic cameras or depth cameras. The SLAM solution was built by significantly modifying and extending a recent

open-source SLAM solution that is fundamentally different from a traditional approach to solving SLAM problem. The modifications made are those needed to provide the position measurements necessary for the navigation solution on a UAV while simultaneously building the map, all while maintaining control of the UAV. The main contributions of this part include: 1) extension of the map building algorithm to enable it to be used realistically while controlling a UAV and simultaneously building the map; 2) improved performance of the SLAM algorithm for lower camera frame rates; and 3) the first known demonstration of a monocular SLAM algorithm successfully controlling a UAV while simultaneously building the map. This work demonstrates that a fully autonomous UAV that uses monocular vision for navigation is feasible, and can be effective in Global Positioning System denied environments.

MONOCULAR VISION BASED LOCALIZATION AND MAPPING

by

MICHAL JAMA

M.S., Wroclaw University of Technology, Poland, 2007

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2011

Approved by:

Co-Major Professor
Balasubramaniam Natarajan

Approved by:

Co-Major Professor
Dale Schinstock

Copyright

Abstract

In this dissertation, two applications related to vision-based localization and mapping are considered: (1) improving navigation system based satellite location estimates by using on-board camera images, and (2) deriving position information from video stream and using it to aid an auto-pilot of an unmanned aerial vehicle (UAV).

In the first part of this dissertation, a method for analyzing a minimization process called bundle adjustment (BA) used in stereo imagery based 3D terrain reconstruction to refine estimates of camera poses (positions and orientations) is presented. In particular, imagery obtained with pushbroom cameras is of interest. This work proposes a method to identify cases in which BA does not work as intended, i.e., the cases in which the pose estimates returned by the BA are not more accurate than estimates provided by a satellite navigation systems due to the existence of degrees of freedom (DOF) in BA. Use of inaccurate pose estimates causes warping and scaling effects in the reconstructed terrain and prevents the terrain from being used in scientific analysis. Main contributions of this part of work include: 1) formulation of a method for detecting DOF in the BA; and 2) identifying that two camera geometries commonly used to obtain stereo imagery have DOF. Also, this part presents results demonstrating that avoidance of the DOF can give significant accuracy gains in aerial imagery.

The second part of this dissertation proposes a vision based system for UAV navigation. This is a monocular vision based simultaneous localization and mapping (SLAM) system, which measures the position and orientation of the camera and builds a map of the environment using a video-stream from a single camera. This is different from common SLAM solutions that use sensors that measure depth, like LIDAR, stereoscopic cameras or depth cameras. The SLAM solution was built by significantly modifying and extending a recent

open-source SLAM solution that is fundamentally different from a traditional approach to solving SLAM problem. The modifications made are those needed to provide the position measurements necessary for the navigation solution on a UAV while simultaneously building the map, all while maintaining control of the UAV. The main contributions of this part include: 1) extension of the map building algorithm to enable it to be used realistically while controlling a UAV and simultaneously building the map; 2) improved performance of the SLAM algorithm for lower camera frame rates; and 3) the first known demonstration of a monocular SLAM algorithm successfully controlling a UAV while simultaneously building the map. This work demonstrates that a fully autonomous UAV that uses monocular vision for navigation is feasible, and can be effective in Global Positioning System denied environments.

Table of Contents

List of Figures

List of Tables

Glossary of Terms and Symbols

GPS	Global Positioning System
$MOLA$	Mars Orbiter Laser Altimeter
$HiRISE$	High Resolution Stereo Camera
$LROC$	Lunar Reconnaissance Orbiter Camera
$PTAM$	Parallel Tracking and Mapping
$PTAMM$	Parallel Tracking and Multiple Mapping
LoG	Laplacian of Gaussian
GPU	graphical processing unit
DEM	digital elevation model
DTM	digital terrain model
BA	bundle adjustment
ED	ephemeris data
FOV	field of view
GCP	ground control point
DOF	degree of freedom
ML	maximum likelihood
UAV	unmanned aerial vehicle
$VTOL$	vertical take-off and landing
$SLAM$	simultaneous localization and mapping

Acknowledgments

First, I would like to express my utmost gratitude to my three mentors: Dr. Chris Lewis, Dr. Bala Natarajan and Dr. Dale Schinstock. Their continues support, words of encouragement and faith in me, allowed me to overcome my weaknesses and carry out this research.

I would like to thank Dr. Dwight Day, Dr. Guoqiang Hu and Dr. Michael O'Shea for serving on my committee and for their valuable suggestions about this work.

A special thanks to all my office mates and friends in Manhattan, KS and around the world. Our conversations and time we spent together helped me to stay sane.

Most, I would like to thank my whole family for their unshakable faith and trust in me. Thanks for always being there for me and for all your love.

Lastly, I would like to thank my wife Magdalena. Thanks for all support and understanding that I needed so much.

Dedication

To my family and friends.

Chapter 1

Introduction

In this dissertation, we look at two applications where precise location information is needed. In the first one, we study the problem of improving satellite location estimates obtained by the navigation systems by using images captured by the satellites. In the second application we propose an algorithm that derives position information from video and feeds that knowledge into the auto-pilot system of an unmanned aerial vehicle (UAV). This work in no way pretends to be a guide to all methods used in the field. Rather, it focuses on these two specific problems and offers strategies to enhance their solutions.

The creation and public availability of a Global Positioning System (GPS) irreversibly changed the way we think about traveling. This *localization* technology has transformed the way transportation industry operates and the introduction of affordable personal GPS receivers has opened a door for service providers to deliver location dependent content to the user. GPS derived location information makes the work of airplane pilots, ship captains or car drivers easier. Additionally, many autonomous systems require it to operate properly. However, sometimes the GPS location service may not provide the accuracy needed while at other times, it may be unavailable, e.g., indoors or in a battle field where GPS signal is jammed. Moreover, GPS provides only position estimates, while in some applications, like autonomous UAV navigation, it is also required to provide orientation information. Since knowing a precise location is desirable in many applications, researchers have developed various methods for obtaining this information based on non-GPS sensors.

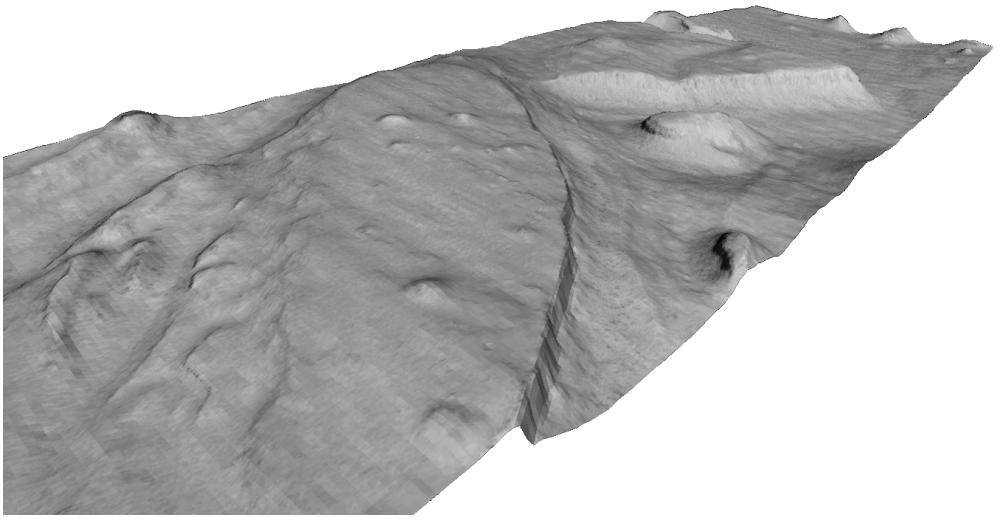


Figure 1.1: *Example of a Digital Terrain Model (DEM) reconstructed from a stereo pair of images by using Ames Stereo Pipeline, originally appeared in [1]*

In the next section, we provide a background to localization and give an overview of some of applications that benefit from location information. Next, we present a literature review of solutions related to the two problems of interest. We then describe the key contributions of this work, and conclude with the dissertation overview.

1.1 Background

This work is concerned with localization and mapping based on a visual input (camera images). By localization, we mean estimation of an object pose (position and orientation) with respect to the environment, while by mapping, we mean a process of collecting information about the environment. More formally, localization is an estimation of an object pose given a set of landmarks with known positions that are observable by the object. Mapping is a process of building a set of landmarks - a map - and estimating their position given the pose of the object that observes these landmarks. Those two processes are coupled together. To solve for the object pose, an a-priori knowledge of the map is required. To solve for the map, an a-priori knowledge of the object pose is needed. The solution to this, which

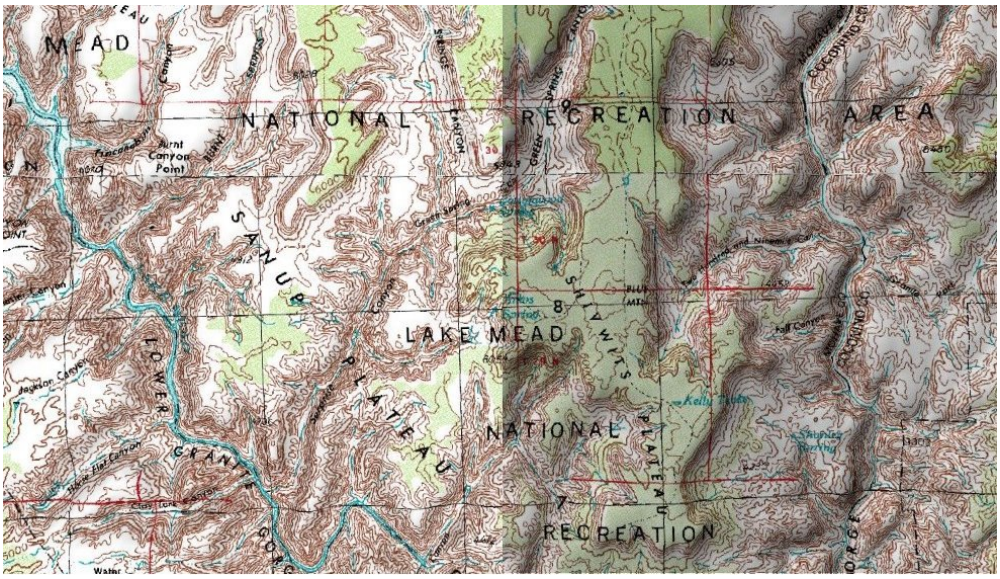


Figure 1.2: *Comparison of a normal map (left) to a shaded terrain relief (right).*

seems to be a "chicken and an egg" problem, can be obtained when the problems are solved together. The combined problem is called simultaneous localization and mapping (SLAM) in the literature.

The name SLAM appeared for the first time in the 1990s in the robotics community. Ability to place a robot in an unknown environment and use it to incrementally build the map of the environment while localizing itself within this map was considered the "holy grail" [3, 4]. Solving the SLAM problem provided one of the enabling technologies needed for a fully autonomous system. Military applications of such systems are a primary target. Additionally, many robotic systems have been proposed to help with search and rescue missions or emergency response [5–8]. Solution of the vision based SLAM has also been applied outside of the robotics domain. Augmented reality solutions use SLAM to track camera position and place virtual objects into the map. Proposed applications include personal touring systems for museums and exhibitions or interactive tutorials on using equipment [9, 10].

Some applications require the location solution to be available in real time (e.g., for robot or UAV navigation) while others may need to obtain the solution as a post processing

step of the data collection (e.g., in structure from motion). The first part of this dissertation looks at the properties of an off-line processing of satellite imagery to improve the ephemeris data (measurements provided by a navigation system of the satellite). The second part uses the SLAM algorithm to process video input in real time and provide navigation data for a UAV.

Satellite and aerial imagery has been used to provide a variety of products, including digital elevation models (DEMs) (cf. Fig. 1.1), orthorectified images, or shaded reliefs (cf. Fig. 1.2). In order to create the last two products, a DEM needs to be available. DEM can be reconstructed by using a stereo-pair of images. First mapping between pixels in both images needs to be established and then a terrain model can be reconstructed based on this mapping and set of camera poses by using triangulation. Camera pose measurements are provided by an onboard navigation system and usually contain a significant error. At the same time geometric information contained within the captured images is of much higher quality and can be used to improve camera pose estimates. A process of improving these pose estimates by using features identified in both images is called bundle adjustment (BA) [11–13]. BA is a minimization problem that, given a set of points identified in multiple images, tries to simultaneously estimate those points' 3D locations and camera poses used to obtain the images. Application of BA to imagery coming from an extraterrestrial mission like Mars Reconnaissance Orbiter [14, 15] or Lunar Reconnaissance Orbiter [16, 17] is of primary interest in the first part of this work.

In the minimization formulated by BA, camera poses and 3D locations of tie points are independent variables . In most cases, if the process is given enough number of matched pixels, the minimization in BA has a single global minimum, that the process converges to. However, as shown later in this work, under some conditions, the minimization can have infinite number of global minimums that correspond to significantly different estimates of the independent variables. In other words, there may exist degree(s) of freedom (DOF) that prevents the BA from improving estimates of some of the independent variables, e.g., the

camera poses.

1.2 Related Work and Motivation

1.2.1 Bundle Adjustment

The concept of BA was first introduced within the photogrammetry field [18, 19] and it has been documented by many textbooks [11, 20–22]. More recently, close-range photogrammetry gained popularity within the computer vision community [12, 13]. Lourakis and Argyros released an open-source implementation of a sparse BA to the community[23]. In [24], they compare different minimization techniques suited for BA.

Li et al. describe a mathematical model for a pushbroom camera and performed accuracy analysis using simulated images [25]. The works of Poli [26, 27], study pushbroom camera image acquisition models with an emphasis on different representations of the path. In [28], Shan et al. analyze the photogrammetric mapping properties and capabilities of the Mars Global Surveyor. Stereo images from the Mars Orbiter Camera are used to generate DEMs which are registered to Mars Orbiter Laser Altimeter data. Spiegel et al. also use MOLA to register DEMs constructed with images from the High Resolution Stereo Camera [29]. The authors in [30] present an approach for processing HiRISE images where BA was used as a step before DEM extraction. Hwangbo et al. extend this approach by incorporating the Mars Orbiter Laser Altimeter into BA [31].

1.2.2 Simultaneous Localization and Mapping

Detailed introduction to SLAM history, formulation and existing algorithms is provided by Durrant-Whyte and Bailey [3, 4]. Their papers describe standard probabilistic approach to SLAM and mentions two dominant frameworks for solving it: the Extended Kalman Filter and the Rao-Blackwellized filter. Bryson and Sukkarieh presents a path planning algorithm for a UAV that employs a SLAM algorithm for localization [32]. Their path planner considers the requirements of SLAM algorithms, namely visibility of features in the

images being tracked and the trade-off between exploration of the scene and accuracy of the localization. In [33], Sazdovski and Silson describe a system where IMU measurements are combined with video-based feature measurement in the SLAM algorithm. However, only simulation experiments are presented by the authors, in [33].

The work of Kim and Sukkarieh [34], demonstrates a system where a UAV is used to capture video and IMU data that are processed off-line using a Kalman based SLAM algorithm. Artieda et al. use a UAV to capture video data that is processed off line with a Kalman based SLAM algorithm [35]. SLAM position solutions are compared to GPS measurements. Another paper by Caballero et al. describes use of the Extended Kalman filter-based SLAM to compute the localization and mapping of a UAV [36]. Data are recorded and processed off-line. The sFly group uses a SLAM algorithm to perform a real-time UAV navigation [37–39]. Their approach utilizes an off-board processing and wired connection. However, it proves the validity of an idea of controlling a quad based on video navigation. In [40], Bachrach et al. present solution for autonomous UAV flights in indoor environments. However, their solution is based on a LIDAR scanner and not on a vision-based algorithm. Mellinger et al. present a control system where aggressive maneuvers of UAVs are possible [41]. The control system relies on a VICON motion capture system which uses multiple external cameras to determine the position of the UAV.

Work presented by Klein and Murray gives a novel approach to SLAM wherein bundle adjustment is used instead of filters [42]. The algorithm accuracy and robustness of their design are superior relative to any known real-time SLAM algorithm based on filtering. A key paper by Strasdat et al. compares performance of SLAM algorithms based on filtering and the SLAM algorithm based on bundle adjustment. Conclusions of the paper clearly state that the number of features that the algorithm is able to use has far more influence on accuracy than the number of measurements provided for a single feature, therefore, favoring bundle adjustment [43]. Recently some authors showed that by using PTAM results and performing optical flow computation on graphical processing units (GPUs), it is possible

to perform dense scene reconstruction in a real time [44–46]. SLAM algorithms are also applied to non robotic problems. In [47], Mountney et al. use SLAM algorithm to develop a system aiding surgeons during minimally invasive surgery.

1.2.3 Motivation

HiRISE and LROC cameras have been providing images of Mars and the Moon with unprecedented quality. Some of those images are stereo-pairs which can be used in DEM generation. This provides an opportunity for scientists to study the surface, geological processes, and climate changes on Mars and the Moon. However, current stereo processing capabilities are very limited, mostly due to the manual labor that is required to process a stereo-pair. Extending these capabilities, especially towards an automated system, is a motivation for the research presented within this dissertation.

The second part of the effort stems from the rapid evolution of the SLAM algorithms. Improved accuracy, robustness, and real-time processing enables the use of this technology to provide a navigation solution for UAVs. Autonomous UAVs can be used in emergency response or in search and rescue missions. In these type of missions, there are often health risks associated with a human presence on-site which can be minimized by using robots. Additionally, these missions usually have to operate in a GPS-denied environments, motivating the need for a different positioning and navigation strategy. Vision based navigation solution provides an affordable option to address this need. Furthermore, our engagement in OpenPilot, an open-source and open-hardware project developing an autopilot system for UAVs, makes us believe that our SLAM solution can be applied by many users worldwide and have a significant impact.

1.3 Contributions

The major contributions of this work are presented below. With regards of the first problem of interest, namely in improving satellite location estimates obtained via the inertial

navigation systems by using images captured by the satellites:

- We propose a mathematical method for finding DOF in BA minimization.
- The proposed method is applied to analyze common satellite geometries used to capture the imagery. We demonstrate that for two geometries DOF exist and the BA does not have a unique solution. This implies that BA pose estimates may be extremely inaccurate without BA providing any indication about the magnitude of the error.
- We expose deleterious effects of DOF on scene reconstruction using real data from the Mars Reconnaissance Orbiter [14, 15]. This result postulates cautiousness when inferring conclusions based on terrain reconstructions obtained with pushbroom imagery.
- We show that, for aerial imagery, a significant improvement can be achieved by carefully considering scene geometries.

Detailed analysis are presented in Chapter 3 and are also presented in our papers [48–50].

Contributions with regards to the second problem of interest i.e., on providing an algorithm for vision only based UAV navigation:

- A state of the art SLAM algorithm is developed, tested and validated by modifying BA based SLAM algorithm.
- We demonstrate using experiments with a hardware prototype, that airframe navigation is possible by means of a vision-only based algorithm.
- Up to our knowledge, it is the first demonstration of a monocular SLAM algorithm successfully controlling a UAV while simultaneously building the map.
- Our results and SLAM solution is a significant addition to "OpenPilot" [51], an open-source project that is available to research and development community.

Our approach is described in Chapter 4 and also in [52]

1.4 Organization of the dissertation

This dissertation is structured into 5 chapters. Chapter 2 presents fundamental methods and algorithms in the minimization, feature detection and matching needed as a background for further chapters. BA formulation is also given in Chapter 2. Chapter 3 presents the derivation of the method for detecting DOF, followed by detailed analysis. Chapter 4 describes development and extensions of the video-based tracking algorithm for UAV navigation and also presents results of UAV under control of the developed algorithm. Finally, Chapter 5 summarizes this dissertation and possible future extensions of this work.

Chapter 2

Localization and Mapping – Fundamentals

This chapter provides a background for the work presented in the chapters following it. Algorithms described here play an important role in the solutions presented later. First feature detectors and descriptors are presented. Next a set of basic methods for non-linear minimization is described. Following is a description of a minimization process called bundle adjustment (BA). BA is commonly used to improve camera pose estimates when dealing with scene reconstruction from images.

2.1 Feature Detection and Matching

This work uses methods for automated detection of distinguishable features in an image. If such features are found on two or more images they can be matched to create an image correspondence. Such correspondence can be used to e.g. stitch images together, detect similar objects or seed an algorithm to find dense image correspondence, i.e., a correspondence where each pixel in one image has an assigned location in the other image. Additionally if camera poses are known matched features can be triangulated to create a three dimensional map of the scene visible in the images. Such features are commonly used in real-time tracking of the camera motions.

The detection and matching process is better described when it is split into three parts:

- Feature detection
- Feature description
- Matching

Feature detection and feature description operate on single images. Matching operates on feature descriptors for two or more images generated in the previous step.

Feature detection locates pixels in the image that are distinguishable, usually corresponding to corners or transitions between objects in the underlying scene. Usually such interesting pixels have very different intensity than most of the pixels around it. Therefore methods for feature detection employ some kind of high pass filtering of the image. It is also very common to use non-maximal suppression on the found pixels to return only the strongest features and prevent detecting the same feature multiple times.

Feature description generates a universal description for detected features. Description should be as independent as possible from rotation or lighting conditions under which a feature was seen in the image to allow feature identification across different views. Using image gradients or wavelet responses computed for small a grid in the neighborhood of the feature is a popular choice here. Responses for each element of the grid are oriented with respect to the dominant response (for rotation invariance) and stored as a descriptor of a feature.

Feature matching finds pairs of descriptors generated from two images that are very close to each other, i.e., they describe the same feature. Correlation is used to compare descriptors and use of KD-Trees enables efficient search for the nearest neighbor.

Early work on feature detectors can be attributed back to Moravec's work on stereo matching using corner detectors [53]. Moravec's method uses differences of image intensities computed along four directions to detect variance in image intensity and mark corners. Moravec's algorithm was improved my Harris and Stephens by using gradients computed in all directions to allow detection of arbitrarily rotated corners [54]. This modified algorithm,

so called Harris corner detector, allowed for rotation invariant detection and gained a wide popularity within the image processing community. A similar approach was proposed by Shi and Thomasi [55]. However, these algorithms are not scale invariant and, therefore, do not allow matching of features seen at different scales.

The groundbreaking research for the creation of scale invariant descriptors was conducted by Lindeberg by describing scale-space theory [56–58]. He proposed a solution to the problem of identifying an appropriate and consistent scale for feature detection based on using a response of a Laplacian of Gaussian (LoG) of the image to detect features. Where the LoG changes sign is a strong indication of an edge. The edges must be searched further to find corners.

The development of the first highly accurate and repeatable feature detector accompanied by the scale and rotation invariant feature descriptor called SIFT is due to Lowe [59]. It uses a Difference of Guassians (DoG) to approximate the LoG. Shortly after, an algorithm called SURF by Bay et al. used integral images to speed the calculation of the Hessian (second derivative) at each pixel to detect corners directly rather than searching for edges [60].

This group of algorithms can be characterized as corner response based detectors. All of the mentioned algorithms compute the image response for variously defined corner-detecting functions. When the function value for a pixel is above a threshold and it is a local maximum the pixel is identified as a feature. A different approach is used by the next two algorithms.

The SUSAN detector was proposed by Smith and Brady [61]. SUSAN identifies corners and edges by analyzing the number of pixels in the area around interest point which have a similar intensity to the interest point. FAST algorithm proposed by Rosten and Drummond goes a step further [62]. It only investigates the pixel on the edge of a circle surrounding the interest point.

2.2 Non-linear minimization

In this section we present basic methods used to find a minimum of a non-linear function. Derivations of Newton, Gauss-Newton and gradient descent are presented first. Next Levenberg–Marquardt method is described. The development follows the ones presented in [13, 63]

The problem is stated as follows. We are given a function $C(\mathbf{K})$ that we want to minimize where $\mathbf{K}$ is a parameter vector that we can vary. Since $C(\mathbf{K})$ may not be linear we use iterative strategy to obtain a solution. We want to start with an estimate of a solution, $\mathbf{K}_0$, and at each step look for $\mathbf{K}_i = \mathbf{K}_{i-1} + \delta_{\mathbf{K}}$ for which $C(\mathbf{K}_i) < C(\mathbf{K}_{i-1})$. Four approaches that compute step $\delta_{\mathbf{K}}$ are presented below.

Newton method

We can use a truncated Taylor series to expand $C(\mathbf{K})$

$$C(\mathbf{K} + \delta_{\mathbf{K}}) \approx C(\mathbf{K}) + \mathbf{g}\delta_{\mathbf{K}} + \frac{1}{2}\delta_{\mathbf{K}}^T \mathbf{G} \delta_{\mathbf{K}} \tag{2.1}$$

where $\mathbf{g}$ is the gradient, $\mathbf{G}$ is a Hessian and functions are evaluated at $\mathbf{K}$. Higher than second order terms were omitted in the above equation. Now to minimize $C(\mathbf{K})$ we take derivative of the right hand side of Eq. 2.1. Using the product rule it can be shown [63] that

$$\nabla \left(\mathbf{u}^T \mathbf{v}\right) = \left(\nabla \mathbf{u}^T\right) \mathbf{v} + \left(\nabla \mathbf{v}^T\right) \mathbf{u} \tag{2.2}$$

Therefore we have

$$\frac{1}{2}\left(\mathbf{G} + \mathbf{G}^T\right)\delta_{\mathbf{K}} + \mathbf{g} = \mathbf{G}\delta_{\mathbf{K}} + \mathbf{g}$$

where the last equality follows from the symmetry of Hessian. Finally setting it to zero

$$\mathbf{G}\delta_{\mathbf{K}} = -\mathbf{g} \tag{2.3}$$

The last equation can be solved for $\delta_{\mathbf{K}}$ using e.g. LDL decomposition. However, the main drawback of the method is the requirement of computing a Hessian which is time consuming. The following methods improve on that by constraining the form of $C(\mathbf{K})$ and using approximation schemes.

Gauss-Newton method

From now on, lets assume that the $C(\mathbf{K})$ is in the form of sum of squared terms

$$C(\mathbf{K}) = \mathbf{c}^T\mathbf{c} \tag{2.4}$$

where $\mathbf{c} = \mathbf{c}(\mathbf{K})$. This is so-called non-linear least squares problem, for which an efficient approximation of the Newton step Eq. 2.3 can be computed. Under this form, using Eq. 2.2, the gradient $\mathbf{g}$ becomes

$$\mathbf{g} = 2\left(\frac{\partial \mathbf{c}}{\partial \mathbf{K}}\right)^T \mathbf{c} = 2\mathbf{J}^T\mathbf{c} \tag{2.5}$$

and Hessian $\mathbf{G}$ becomes

$$\begin{aligned}\mathbf{G} &= 2\left(\frac{\partial \mathbf{c}}{\partial \mathbf{K}}\right)^T \frac{\partial \mathbf{c}}{\partial \mathbf{K}} + 2\left(\frac{\partial^2 \mathbf{c}}{\partial \mathbf{K}\partial \mathbf{K}}\right)^T \mathbf{c} \\ &= 2\mathbf{J}^T\mathbf{J} + 2\sum_{i=1}^{m} \mathbf{c}\nabla^2\mathbf{c} \qquad (2.6)\\ &\approx 2\mathbf{J}^T\mathbf{J} \qquad (2.7)\end{aligned}$$

The last approximation follows from the assumption that elements of $\mathbf{c}$ are small and the whole term can be neglected. Plugging those expressions back to Eq. 2.3 we obtain

$$\mathbf{J}^T\mathbf{J}\delta_{\mathbf{K}} = -\mathbf{J}^T\mathbf{c} \tag{2.8}$$

This again can be solved for $\delta_{\mathbf{K}}$. However, this time there is no need to compute the Hessian which is approximated by the Jacobian.

Gradient descent

For the sum of squares function the gradient is given by Eq. 2.5. Another method for finding $\delta_{\mathbf{K}}$ is to simply choose it to move in the negative direction of the gradient, i.e., in the direction of the steepest descent. The step is given as

$$\lambda\delta_{\mathbf{K}} = -\mathbf{J}^T\mathbf{c} \tag{2.9}$$

where λ is a scaling factor that determines the length of the step and can be determined for each iteration independently using line search or trust region.

Levenberg - Marquardt method

This method combines Gauss–Newton with the gradient descent. The iteration step is given as

$$\left(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}\right)\delta_{\mathbf{K}} = -\mathbf{J}^T\mathbf{c} \tag{2.10}$$

The value of λ varies for each iteration. If for an $i-th$ iteration $C(\mathbf{K}_i) < C(\mathbf{K}_{i-1})$ holds, then λ is decreased by a factor (usually 10), otherwise the lambda is increased by a factor. After looking at Eq. 2.10 one can notice that when λ is big, $\mathbf{J}^T\mathbf{J}+\lambda\mathbf{I}$ is dominated by the $\lambda\mathbf{I}$ part and the equation degrades to gradient descent (c.f. Eq. 2.9). On the other hand when λ is small then $\mathbf{J}^T\mathbf{J}+\lambda\mathbf{I}$ is dominated by $\mathbf{J}^T\mathbf{J}$ and the equation becomes the Gauss–Newton step (c.f. Eq. 2.8). Therefore the method switches to gradient descent search when it has troubles finding a good step $\delta_{\mathbf{K}}$ using Gauss–Newton. On the other hand when updates $\delta_{\mathbf{K}}$ decrease the cost it uses Gauss-Newton to speed up the convergence.

If covariance matrix $\mathbf{\Sigma}$ is known for the measurement it can be incorporated into step equation as a weighting matrix

$$\left(\mathbf{J}^T\mathbf{\Sigma}^{-1}\mathbf{J} + \lambda\mathbf{I}\right)\delta_{\mathbf{K}} = -\mathbf{J}^T\mathbf{\Sigma}^{-1}\mathbf{c} \tag{2.11}$$

2.3 Bundle Adjustment

This section provides a formulation of a bundle adjustment (BA) process. BA is commonly used with stereo imagery in terrestrial and exploration applications to refine the estimates of camera pose. The process is often used to improve products such as georeferenced imagery and digital elevation models (DEMs) produced through triangulation of corresponding pixels in stereo pairs. It is also employed in some of the new simultaneous localization and mapping (SLAM) algorithms (e.g. [42]) to maintain accuracy of the environment map. BA is a well known process formulated in the late 1950s by Brown and Schmid [18, 19] that has been extensively used in the photogrammetric community since then [11, 20]. More recently it has also been documented in several texts and extended papers due to its increasing popularity

within the computer vision community [12, 13].

The typical formulation of BA adjusts all scene parameters including the camera positions, $\mathbf{p}_c$, camera attitudes, $\boldsymbol{\phi}_c = [\omega\, \phi\, \kappa]^T$, and feature (tie points) locations in 3D, $\mathbf{p}_f$, to minimize a scalar cost function. The cost function is typically a sum of squared errors, however different formulations were also proposed. Individual error terms for tie points are measured in the camera's focal plane as the difference between measured location of the feature in the image, $\mathbf{p}_m$, and that predicted by the current state of the estimated parameters, $\mathbf{p}_b$. Prediction of a feature's image location is accomplished through back-projection of $\mathbf{p}_f$, into the camera's focal plane. Without lens distortion, the pinhole camera model describes how 3D points are imaged. With this model, the projection of $\mathbf{p}_f$ into the camera focal plane is given by

$$\mathbf{p}_b = \begin{bmatrix} u_b \\ v_b \end{bmatrix} = -\frac{f}{d_z} \begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{2.12}$$

where f is the camera focal length and

$$\mathbf{d} = [d_x\, d_y\, d_z]^T = \mathbf{R}_{cr}(\mathbf{p}_f - \mathbf{p}_c) \tag{2.13}$$

and

$$\mathbf{R}_{cr} = \begin{bmatrix} c_\phi c_\kappa & c_\omega s_\kappa + s_\omega s_\phi c_\kappa & s_\omega s_\kappa - c_\omega s_\phi c_\kappa \\ -c_\phi s_\kappa & c_\omega c_\kappa - s_\omega s_\phi s_\kappa & s_\omega c_\kappa + c_\omega s_\phi s_\kappa \\ s_\phi & -s_\omega c_\phi & c_\omega c_\phi \end{bmatrix}. \tag{2.14}$$

Here $c_* = \cos(*)$ and $s_* = \sin(*)$. The back-projection error between an observed feature, $\mathbf{p}_m$, and its prediction, $\mathbf{p}_b$, is given by

$$\mathbf{e} = \mathbf{p}_m - \mathbf{p}_b = \begin{bmatrix} u_m \\ v_m \end{bmatrix} - \begin{bmatrix} u_b \\ v_b \end{bmatrix} = \begin{bmatrix} \delta u \\ \delta v \end{bmatrix}. \tag{2.15}$$

The scalar cost function measuring the scene consistency is the sum of squared back projection errors,

$$C = \mathbf{c}^T \mathbf{c}, \tag{2.16}$$

where, for m images with n tie points, the cost vector is

$$\mathbf{c} = [\mathbf{e}_1^T \ \ldots \ \mathbf{e}_m^T]^T \tag{2.17}$$

where

$$\mathbf{e}_i = [\mathbf{e}_{1,i}^T \ \dots \ \mathbf{e}_{n,i}^T]^T \quad \text{for} \quad i = 1 \dots m. \tag{2.18}$$

Here, for simplicity, it is assumed that all n tie points are observed in all m cameras. For n tie points and m cameras, if we define the vector of parameters to be estimated as

$$\mathbf{k} = [\mathbf{p}_{f_1}^T \ \dots \ \mathbf{p}_{f_n}^T \ \mathbf{p}_{c_1}^T \ \boldsymbol{\phi}_{c_1}^T \ \dots \ \mathbf{p}_{c_m}^T \ \boldsymbol{\phi}_{c_m}^T]^T, \tag{2.19}$$

then plugging Eq. 2.12 into Eq. 2.15 and the result into Eq. 2.17 provides a scalar cost as a function of the scene parameters, $C = C(\mathbf{k})$.

Adding additional constraints such as GCPs to BA is achieved by adding elements to the cost vector that are the difference between a measured value and the value predicted using the parameter vector, similar to Eq. 2.15 as follows

$$\mathbf{c} = [\mathbf{e}_1^T, \dots, \mathbf{e}_m^T, \mathbf{e}_{GCP}^T, \mathbf{e}_{ED}^T]^T, \tag{2.20}$$

where $\mathbf{e}_{GCP}$ are error vectors associated with GCPs.

Ground control points have different variance than back-projection errors, therefore a weighting matrix is defined using the inverse covariance to adjust for their relative significance. The weighting matrix, $\mathbf{W}$, is diagonal assuming no correlation between the errors in individual measurements. The scalar cost then becomes,

$$C = \mathbf{c}^T \mathbf{W} \mathbf{c}. \tag{2.21}$$

The cost function, presented in Eq. 2.21, is non-linear w.r.t. the vector of parameters $\mathbf{k}$. Therefore an iterative algorithm is used to find a $\mathbf{k}$ that minimizes the cost function. The standard approach for BA utilizes the Levenberg-Marquardt (LM) algorithm which is a hybrid between gradient descent and Newton methods. At each update step the incremental adjustment to $\mathbf{k}$ in the LM algorithm is computed using

$$\delta_{\mathbf{k}} = -(\mathbf{J}^T \mathbf{W} \mathbf{J} + \lambda \mathbf{I})^{-1} \mathbf{J}^T \mathbf{W} \mathbf{c}$$

where

$$\mathbf{J} = \frac{\partial \mathbf{c}}{\partial \mathbf{K}} \tag{2.22}$$

is the Jacobian matrix and λ is the LM damping factor, which varies during minimization and tends to zero as a minimum is approached.

In summary, BA is an iterative algorithm that seeks to minimize the cost function $C(\mathbf{k})$ by varying elements of the parameter vector $\mathbf{k}$. Because the camera pose and 3D feature coordinates are elements of $\mathbf{k}$, BA can be conceptualized as simultaneous adjustment of camera positions and feature coordinates to minimize the difference between the measured feature coordinates in the images and those predicted by projecting the 3D coordinates into the focal planes of the cameras.

2.3.1 Different cost functions

In the presented formulation the used cost function is a least–squares estimator. Therefore no optimality can be claimed by this estimator. It is interesting to investigate different formulations of the cost function and possibly design a better one with respect to some measures. Poor [64] points to Maximum Likelihood (ML) estimator as first candidate because of its desired asymptotic properties and relatively easy derivation.

For the following derivations it is easier to look at the image features locations from a different perspective that the one given by Eq. 2.15. To simplify notation we drop the distinction for vertical and horizontal components of a vector $\mathbf{p}_m$ and we treat both components the same way. This is justified as we expect the noise to be equal in both directions. If we assume that no noise is present and there are no distortions an $i-th$ point would be seen in the image at $\mathbf{p}_{b_i} = [u_{b_{j-1}}, u_{b_j}]^T$ where $j = 2i$. At the presence of the noise the $i-th$ point would be observed at $\mathbf{p}_{m_i} = [u_{m_{j-1}}, u_{m_j}]^T$ In the following discussion we assume no lens distortion. The reason for that is that usually we know camera calibration data and using them we can correct optical distortions. Moreover distortions can be assumed constant (at least throughout image acquisition time) and would merely account for a shift

in $\mathbf{p}_m$. Therefore the main source of inaccuracy is noise.

Maximum Likelihood estimator under the Gaussian noise

Given the model above, first lets assume that the noise is Gaussian distributed. Then a single point projection is Gaussian distributed as well with the mean $\mathbf{p}_{m_i}$ and the variance σ^2. Moreover all projections are jointly Gaussian distributed with the mean $\mathbf{p}_0 = \left[{\mathbf{p}_{b_1}}^T, \ldots, {\mathbf{p}_{b_n}}^T\right]^T$ and the covariance $\mathbf{\Sigma}$. Therefore we can write

$$\omega(\mathbf{p}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{\Sigma}|} \exp\left\{-(\mathbf{p}-\mathbf{p}_0)^T \mathbf{\Sigma}^{-1} (\mathbf{p}-\mathbf{p}_0)\right\}$$

where $\mathbf{p} = \left[{\mathbf{p}_{m_1}}^T, \ldots, {\mathbf{p}_{m_n}}^T\right]^T$. Finding ML estimator is equivalent to finding maximum of the above expression

$$\arg\max_{\mathbf{K}} \omega(\mathbf{p}) = \arg\min_{\mathbf{K}} (\mathbf{p}-\mathbf{p}_0)^T \mathbf{\Sigma}^{-1} (\mathbf{p}-\mathbf{p}_0)$$

Since $\mathbf{p}$ is non-linear function of $\mathbf{K}$ we can not find $\mathbf{K}$ directly and have to use some iterative algorithm. Therefore the cost function is given as

$$C = (\mathbf{p}-\mathbf{p}_0)^T \mathbf{\Sigma}^{-1} (\mathbf{p}-\mathbf{p}_0) \tag{2.23}$$

Equation (2.23) is the standard BA cost function. Therefore for BA problem the least–squares and the ML estimator under assumption of a Gaussian noise are the same [65].

Maximum Likelihood under Cauchy noise

We can also derive ML estimator for heavy tailed Cauchy noise. For a single component j of $\mathbf{p}$ the posterior distribution is given as:

$$\omega_j(u_{m_j}) = \frac{b}{\pi}\left[\left(u_{m_j} - u_{b_j}\right)^2 + b^2\right]^{-1}$$

Moreover if we assume that the noise is independent and identically distributed (IID), the joint distribution is given as

$$\omega(\mathbf{p}) = \frac{b^n}{\pi^n} \prod_{j=1}^{2n} \left[\left(u_{m_j} - u_{b_j}\right)^2 + b^2\right]^{-1}$$

Again to find ML estimator we need to maximize above expression

$$\arg\max_{\mathbf{K}} \omega(\mathbf{p}) = \arg\min_{\mathbf{K}} \sum_{j=1}^{2n} \log\left[\left(u_{m_j} - u_{b_j}\right)^2 + b^2\right] \tag{2.24}$$

Therefore

$$C = \sum_{j=1}^{2n} \log\left[\left(u_{m_j} - u_{b_j}\right)^2 + b^2\right] \tag{2.25}$$

Blake–Zisserman

Another cost function was presented in [66]. The distribution function was chosen to represent Gaussian distribution for inliers and uniform distribution for outliers.

$$\omega_j(u_{m_j}) = \exp\left\{-\left(u_{m_j} - u_{b_j}\right)^2\right\} + \epsilon$$

Note that above function is not a true *pdf* since it integrates to infinity. Nevertheless if we use it and like in previous cases solve for the ML estimator we will get

$$C = -\sum_{j=1}^{2n} \log\left[\exp\left\{-\left(u_{m_j} - u_{b_j}\right)^2\right\} + \epsilon\right] \tag{2.26}$$

Huber

This heuristic cost σ function, presented in [13], is based on L1 and least-squares cost functions. For inliers it uses least-squares cost while for outliers it uses L1 cost.

$$C = \begin{cases} \left(u_{m_j} - u_{b_j}\right)^2 & \left|u_{m_j} - u_{b_j}\right| < b \\ 2b\left|u_{m_j} - u_{b_j}\right| & otherwise \end{cases}$$

Function above has only first order continuous derivative. Most iterative methods for finding minimum use first and higher order derivatives therefore its more convenient for implementation reasons to use following approximation (termed *Pseudo-Huber*)

$$C = 2b^2 \sum_{j=1}^{2n} \sqrt{1 + \frac{\left(u_{m_j} - u_{b_j}\right)^2}{b^2}} - 1 \tag{2.27}$$

This approximation has all continuous derivatives.

2.3.2 Bundle adjustment with pushbroom cameras

Presented BA formulation assumes use of frame camera. A camera where the whole image is acquired at the same time from a single pose. Most cameras in popular use are frame cameras, e.g. analog/digital cameras, webcams or video cameras. However, most of the current satellite systems use pushbroom cameras for image acquisition. A pushbroom camera, also called a line scan camera, is an example of a multi-perspective camera. It acquires an image as it travels through space. A pushbroom camera can be thought of as a $1 \times N$ CCD stripe that collects a single scan line at each point along its path (c.f. Fig. 2.1). This results in images without perspective in the along-track direction. Its images have different perspective for each line making processing more challenging [67]. Pushbroom cameras are used on satellites instead of frame cameras because they offer very high signal to noise ratios, (SNR). High SNR allows collection of data even in poor atmospheric conditions.

In order to account for that BA formulation is changed. Unlike frame cameras which capture an entire image from a single pose, pushbroom cameras sweep out images by traveling through space. Each image line is acquired with a slightly different camera pose. Therefore, pose parameters associated with each image are a function of image line number or time, t. In this work we express each camera pose parameter as a nominal function of time plus a time dependent offset. The offset is modeled as a second order polynomial in t whose coefficients are adjusted by BA. The camera pose is given by,

$$\begin{bmatrix} \mathbf{p}_c(t) \\ \boldsymbol{\phi}_c(t) \end{bmatrix} = \begin{bmatrix} x_0(t) + \alpha_x t^2 + \beta_x t + \gamma_x \\ y_0(t) + \alpha_y t^2 + \beta_y t + \gamma_y \\ z_0(t) + \alpha_z t^2 + \beta_z t + \gamma_z \\ \omega_0(t) + \alpha_\omega t^2 + \beta_\omega t + \gamma_\omega \\ \phi_0(t) + \alpha_\phi t^2 + \beta_\phi t + \gamma_\phi \\ \kappa_0(t) + \alpha_\kappa t^2 + \beta_\kappa t + \gamma_\kappa \end{bmatrix} \tag{2.28}$$

where the nominal trajectories $x_0(t)$, $y_0(t)$, $z_0(t)$, $\omega_0(t)$, $\phi_0(t)$ and $\kappa_0(t)$ are assumed to be known measurements provided by a navigation system. The parameter vector Eq. 2.19 is then expanded to include the polynomial coefficients,

$$\mathbf{k} = [\mathbf{p}_{f_1}^T \ \ldots \ \mathbf{p}_{f_n}^T \ [\alpha_{x_1} \ \ldots \gamma_{\kappa_1}] \ \ldots \ [\alpha_{x_m} \ \ldots \gamma_{\kappa_m}]]^T. \tag{2.29}$$

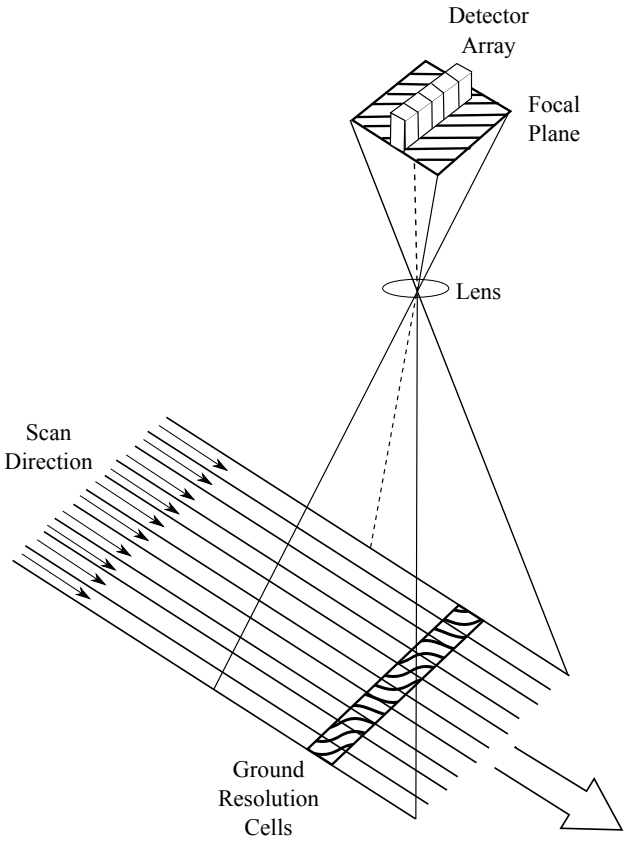


Figure 2.1: *Concept of image acquisition by a pushbroom camera (adapted from [2])*

The necessary addition of these path parameters significantly decreases the constraints on the scene geometry as compared to BA for frame cameras. Additional control measurements, ephemeris data (ED), are commonly included into BA cost vector to add more constraints back. The Eq. 2.20 becomes

$$\mathbf{c} = [\mathbf{e}_1^T, \ldots, \mathbf{e}_m^T, \mathbf{e}_{GCP}^T, \mathbf{e}_{ED}^T]^T, \tag{2.30}$$

where $\mathbf{e}_{ED}$ are error vectors associated with ED.

While additional control measurements can be utilized in BA to help constrain the minimization, the inherent lack of constraints in pushbroom imagery necessitates careful consideration of the geometry used in image acquisition. In the following chapter we will show that this permits the BA optimization process to converge to incorrect solutions by erroneously using the DOF inherent to the viewing geometry.

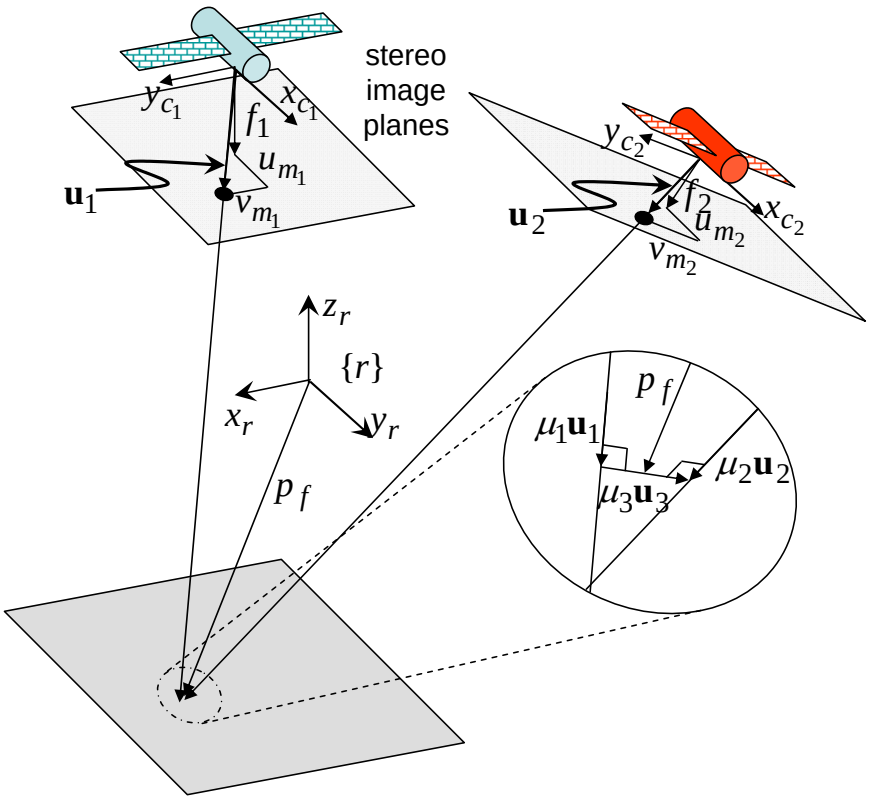


Figure 2.2: *Triangulation using the closest point of approach of two lines*

2.3.3 Different formulation of BA

It is possible to significantly change the definition of the BA in order to e.g. eliminate 3D features from the parameter vector **K** or speed up the processing. Different approaches were described by Moore et al. [68] however none of the new methods was deemed an improvement over a standard formulation. However, some of the methods still proved useful when BA constraints were considered and therefore are presented here.

Elimination of Features from the Parameter Vector - Two Cameras Case

The 3D positions of terrain features can be eliminated from the parameter vector for BA using triangulation. In triangulation the feature positions are found using the camera positions and orientations and the measured image coordinates of the features. If triangulation is formulated as a differentiable function of the other parameters then the feature positions

can be made an implicit result of BA, and removed from the parameter vector.

Several methods exist for triangulation, including BA with the camera positions and orientations held constant, which is an iterative process. However we seek a closed form solution to triangulation. Specifically, the method used here is to take the feature location as the midpoint of the line between the closest points of approach for two lines. The two lines are defined using the camera positions and orientations and the feature image coordinates as shown in Fig. 2.2. The closest points of approach are found by finding the mutual perpendicular for the two lines.

The development of the triangulation equations given here is based on an intuitive geometric description. First two vectors from the focal points of the images to the measured feature locations in the image plane are found using the following equation,

$$\mathbf{u}_i = \mathbf{R}_{rc_i} \begin{bmatrix} \mathbf{p}_{m_i} \\ f_i \end{bmatrix} = \mathbf{R}_{rc_i} \begin{bmatrix} u_{m_i} \\ v_{m_i} \\ f \end{bmatrix}, \quad i = 1, 2 \tag{2.31}$$

where $\mathbf{R}_{rc_i}$ is the transpose of $\mathbf{R}_{c_ir}$ in Eq. 2.14. A mutual perpendicular to $\mathbf{u}_1$ and $\mathbf{u}_2$ is given by the cross product,

$$\mathbf{u}_3 = \mathbf{u}_1 \times \mathbf{u}_2 \tag{2.32}$$

A vector loop can be written that uses the two camera positions, the three pointing vectors, and three to-be-determined scalars (μ_1, μ_2, μ_3) as follows:

$$\mathbf{p}_{c_1} - \mathbf{p}_{c_2} = \mu_1 \mathbf{u}_1 + \mu_3 \mathbf{u}_3 - \mu_2 \mathbf{u}_2 \tag{2.33}$$

The solution of the vector loop Eq. 2.33 for the scalars is found by

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & -\mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix}^{-1} (\mathbf{p}_{c_1} - \mathbf{p}_{c_2}) \tag{2.34}$$

The matrix inversion in Eq. 2.34 will be possible for any reasonable triangulation problem, where $\mathbf{u}_1$ and $\mathbf{u}_2$ are not parallel. In this case the cross product in Eq. 2.32 ensures that the matrix is of full rank. Finally, the triangulated position of the feature is given by taking the

midpoint of the mutual perpendicular

$$\mathbf{p}_f = \mathbf{p}_{c_1} + \mu_1 \mathbf{u}_1 + \mu_3 \mathbf{u}_3/2 \tag{2.35}$$

Plugging Eq. 2.35 into Eq. 2.13 removes the dependency of the cost function on the feature locations, making it only a function of the camera poses and constants. Thus the elements of the parameter vector are reduced to the parameters associated with the camera poses.

Elimination of Features from the Parameter Vector - Three and more cameras

In the case where more than two cameras are used for image acquisition the triangulation method derived in the previous section can not be used. Here a more general triangulation method is described which was similarly presented in [69]. It can be used with an arbitrary number of cameras.

First Eq. 2.13 is rewritten as

$$\begin{gathered}\mathbf{R}_{cr}^T \mathbf{d} = \mathbf{p}_f - \mathbf{p}_c \\ \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^T \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \begin{bmatrix} x_f \\ y_f \\ z_f \end{bmatrix} - \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix},\end{gathered} \tag{2.36}$$

with $r_{i,j}$, x_f, y_f, z_f, x_c, y_c and z_c being defined as components of the respective vectors and matrices in the equation above. Solving Eq. 2.12 for d_x and d_y gives

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = \frac{d_z}{f} \begin{bmatrix} u_m \\ v_m \end{bmatrix}. \tag{2.37}$$

Substituting Eq. 2.37 into Eq. 2.36 gives

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^T \begin{bmatrix} d_z \frac{u_m}{f} \\ d_z \frac{v_m}{f} \\ d_z \end{bmatrix} = \begin{bmatrix} x_f - x_c \\ y_f - y_c \\ z_f - z_c \end{bmatrix}. \tag{2.38}$$

To eliminate d_z the first and second equations in the vector-matrix equation above are

divided by the third to obtain

$$A \equiv \frac{r_{11}\frac{u_m}{f} + r_{21}\frac{v_m}{f} + r_{31}}{r_{13}\frac{u_m}{f} + r_{23}\frac{v_m}{f} + r_{33}} = \frac{x_f - x_c}{z_f - z_c} \tag{2.39}$$

$$B \equiv \frac{r_{12}\frac{u_m}{f} + r_{22}\frac{v_m}{f} + r_{32}}{r_{13}\frac{u_m}{f} + r_{23}\frac{v_m}{f} + r_{33}} = \frac{y_f - y_c}{z_f - z_c}. \tag{2.40}$$

To simplify the notation the left hand sides of the Eq. 2.39 and Eq. 2.40, which contain only known parameters, are replaced by A and B respectively, and these two equations are rewritten in vector matrix form.

$$\begin{bmatrix} 1 & 0 & -A \\ 0 & 1 & -B \end{bmatrix} \begin{bmatrix} x_f \\ y_f \\ z_f \end{bmatrix} = \begin{bmatrix} x_c - z_c A \\ y_c - z_c B \end{bmatrix} \tag{2.41}$$

Both Eq. 2.39 and Eq. 2.40 can be obtained for any feature seen in any view. Therefore for a single feature seen by three cameras we can extend Eq. 2.41 to

$$\begin{bmatrix} 1 & 0 & -A_1 \\ 0 & 1 & -B_1 \\ 1 & 0 & -A_2 \\ 0 & 1 & -B_2 \\ 1 & 0 & -A_3 \\ 0 & 1 & -B_3 \end{bmatrix} \begin{bmatrix} x_f \\ y_f \\ z_f \end{bmatrix} = \begin{bmatrix} x_c - z_c A_1 \\ y_c - z_c B_1 \\ x_c - z_c A_2 \\ y_c - z_c B_2 \\ x_c - z_c A_3 \\ y_c - z_c B_3 \end{bmatrix}. \tag{2.42}$$

This is an over constrained equation of the form $\mathbf{D}\mathbf{p}_f = \mathbf{b}$. To obtain a closed form solution for $\mathbf{p}_f$ the least squares solution is applied,

$$\mathbf{p}_f = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{b}. \tag{2.43}$$

Plugging Eq. 2.43 into Eq. 2.13 removes the dependency of the cost function on the feature locations, making it only a function of the camera EO and constants. Thus the elements of the parameter vector are reduced to the parameters associated with the camera EO.

BA with 3D space based cost element

Another possibility for eliminating features from the parameter vector $\mathbf{K}$ is to eliminate the reason for their presence. Cost elements, similar to Eq. 2.15, can be formulated as a minimum distance between the rays of two cameras pointing to the same feature. Eq. 2.34

provides a solution for μ_3 which is a distance between rays and is used as a cost vector element. Since cost does not require 3D feature positions to be computed features can be dropped from $\mathbf{K}$. However this formulation has many shortcomming, the most serious one for being biased towards smaller scenes. Making the whole scene smaller reduces the distance between the camera rays pointing to the same feature thus reducing the cost.

2.3.4 Efficient BA implementation

General LM implementation can be found in [70]. However, when dealing with minimization problems that use parameter vector with thousands of variables, like in case of BA with multiple cameras and thousands of points, general LM implementation soon becomes intractable. Luckily, BA problem has a very sparse structure and it is beneficial to exploit it in LM formulation. Following discussion is similar to ones presented in [13, 23].

The most time consuming part of the LM is computing the step by solving the Eq. 2.11. We aim to use sparse structure of the problem to represent Eq. 2.11 in a form that does not require solving the whole system of equations simultaneously. It is much more efficient to solve multiple small systems than a single one of much bigger size. The following problem decomposition allows that.

Lets assume that the cost vector $\mathbf{c}$ can be split into n parts

$$\mathbf{c} = \left[\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_n^T\right]^T \tag{2.44}$$

where each $\mathbf{c}_i$ corresponds to all $i-th$ tie point residuals in the images. Moreover lets assume that the parameter vector $\mathbf{K}$ can be divided in similar manner and can be written as

$$\mathbf{K} = \left[\mathbf{a}^T, \mathbf{b}_1^T, \mathbf{b}_2^T, \ldots, \mathbf{b}_n^T\right]^T \tag{2.45}$$

Here $\mathbf{a}$ contains all parameters describing the cameras, while $\mathbf{b}_i$ contains parameters describing an $i-th$ tie point (usually (x, y, z) coordinates). We can observe that each $\mathbf{c}_i$ is dependent only on a single $\mathbf{b}_i$ as each measurement in the image corresponds to a single

tiepoint. Therefore

$$\forall_{i \neq j} \frac{\partial \mathbf{c}_i}{\partial \mathbf{b}_j} = 0 \tag{2.46}$$

By defining

$$\mathbf{A} = \frac{\partial \mathbf{c}}{\partial \mathbf{a}}, \qquad \mathbf{B} = \frac{\partial \mathbf{c}}{\partial \mathbf{b}}$$

and

$$\mathbf{A}_i = \frac{\partial \mathbf{c}_i}{\partial \mathbf{a}}, \qquad \mathbf{B}_i = \frac{\partial \mathbf{c}_i}{\partial \mathbf{b}_i}$$

we can write the Jacobian $\mathbf{J} = \frac{\partial \mathbf{c}}{\partial \mathbf{K}}$ as

$$\begin{aligned} \mathbf{J} &= \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \end{array}\right] \\ &= \left[\begin{array}{c|ccc} \mathbf{A}_1 & \mathbf{B}_1 & & \\ \mathbf{A}_2 & & \mathbf{B}_2 & \\ \vdots & & & \ddots \\ \mathbf{A}_n & & & \mathbf{B}_n \end{array}\right] \end{aligned} \tag{2.47}$$

Assuming that the covariance matrix $\mathbf{\Sigma}$ is block diagonal, i.e., all measurements $\mathbf{c}_i$ are independent with covariance matrices $\mathbf{\Sigma_i}$, we can write

$$\mathbf{J}^T\mathbf{\Sigma}^{-1}\mathbf{J} = \left[\begin{array}{c|ccc} \mathbf{U} & \mathbf{W}_1 & \cdots & \mathbf{W}_n \\ \hline \mathbf{W}_1^T & \mathbf{V}_1 & & \\ \vdots & & \ddots & \\ \mathbf{W}_n^T & & & \mathbf{V}_n \end{array}\right] \qquad \mathbf{J}^T\mathbf{\Sigma}^{-1}\mathbf{c} = \left[\begin{array}{c} \epsilon_{\mathbf{a}} \\ \hline \epsilon_{\mathbf{b}_1} \\ \vdots \\ \epsilon_{\mathbf{b}_n} \end{array}\right] \tag{2.48}$$

where

$$\mathbf{U} = \sum_{i=1}^{n} \mathbf{A}_i^T \Sigma_i^{-1} \mathbf{A}_i, \qquad \mathbf{W}_i = \mathbf{A}_i^T \Sigma_i^{-1} \mathbf{B}_i, \qquad \mathbf{V}_i = \mathbf{B}_1^T \Sigma_1^{-1} \mathbf{B}_1$$

$$\epsilon_{\mathbf{a}} = \sum_{i=1}^{n} \mathbf{A}_i^T \Sigma_i^{-1} \mathbf{c}_i, \qquad \epsilon_{\mathbf{b}_i} = \mathbf{B}_i^T \Sigma_i^{-1} \mathbf{c}_i.$$

Denoting top right and bottom right parts of the first matrix in Eq. 2.48 as $\mathbf{W}$ and $\mathbf{V}$ we express Eq. 2.11 as

$$\left[\begin{array}{cc} \mathbf{U}^* & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V}^* \end{array}\right] \left[\begin{array}{c} \boldsymbol{\delta}_{\mathbf{a}} \\ \boldsymbol{\delta}_{\mathbf{b}} \end{array}\right] = \left[\begin{array}{c} \boldsymbol{\epsilon}_{\mathbf{a}} \\ \boldsymbol{\epsilon}_{\mathbf{b}} \end{array}\right] \tag{2.49}$$

with

$$\mathbf{U}^* = \mathbf{U} + \lambda\mathbf{I} \qquad \mathbf{V}^* = \mathbf{V} + \lambda\mathbf{I} \qquad \boldsymbol{\delta}_\mathbf{K} = [\boldsymbol{\delta}_\mathbf{a}, \boldsymbol{\delta}_\mathbf{b}]^T$$

First we solve for $\boldsymbol{\delta}_\mathbf{a}$. Eq. 2.49 is left multiplied with $\begin{bmatrix} \mathbf{I} & -\mathbf{W}\mathbf{V}^{*-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$ to obtain

$$\begin{bmatrix} \mathbf{U}^* - \mathbf{W}\mathbf{V}^{*-1}\mathbf{W}^T & \mathbf{0} \\ \mathbf{W}^T & \mathbf{V}^* \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_\mathbf{a} \\ \boldsymbol{\delta}_\mathbf{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\epsilon}_\mathbf{a} - \mathbf{W}\mathbf{V}^{*-1}\boldsymbol{\epsilon}_\mathbf{b} \\ \boldsymbol{\epsilon}_\mathbf{b} \end{bmatrix} \tag{2.50}$$

This can be solved for $\boldsymbol{\delta}_\mathbf{a}$ by

$$\left(\mathbf{U}^* - \mathbf{W}\mathbf{V}^{*-1}\mathbf{W}^T\right)\boldsymbol{\delta}_\mathbf{a} = \boldsymbol{\epsilon}_\mathbf{a} - \mathbf{W}\mathbf{V}^{*-1}\boldsymbol{\epsilon}_\mathbf{b}$$

After solving for $\boldsymbol{\delta}_\mathbf{a}$ we can solve for $\boldsymbol{\delta}_\mathbf{b}$ by using back substitution

$$\mathbf{V}^*\boldsymbol{\delta}_\mathbf{b} = \boldsymbol{\epsilon}_\mathbf{b} - \mathbf{W}^T\boldsymbol{\delta}_\mathbf{a}$$

Note that $\mathbf{V}^*$ is block diagonal and its inverse can be simply computed as

$$\mathbf{V}^{*-1} = \begin{bmatrix} \mathbf{V}_1^{*-1} & & \\ & \ddots & \\ & & \mathbf{V}_n^{*-1} \end{bmatrix}$$

Therefore instead of solving one big system for all the tiepoints, here we solve multiple small systems for each tie point separately. This corresponds to reduction in time complexity from $\Theta(n^3)$ needed for solving general system with n equations to roughly $\Theta(n)$ (assuming number of tiepoints parameters is much larger than the number of camera parameters).

2.4 Summary

This chapter presented selected topics that are fundamental to our work presented in the following chapters. Section 2.1 describes feature detection and matching. A process used as a preprocessing step for BA to find tiepoints as well as an initial step of any video based SLAM algorithm. Section 2.2 presents four minimization routines. Newton procedure can be seen as a basic approach, however it needs a second order derivatives (Hessian) to approximate function curvature. This procedure, under special circumstances (function being sum of

squares), can be accurately approximated by a Gauss-Newton method which uses only first order derivatives. Another approach described was gradient descent. The last procedure, a Levenberg-Marquard method, is a hybrid of Gauss-Newton and gradient descent methods. Section 2.3 defines BA problem. Formulation for a frame camera as well as for pushbroom camera is given. Different cost functions are presented. BA efficient implementation is derived that exploit sparse structure of the problem.

Chapter 3

Identifying Degrees of Freedom in Pushbroom Bundle Adjustment

This chapter is concerned with BA for pushbroom cameras whereby pose parameters are refined using two images of the same terrain captured from different vantage points. Pioneering work was performed in this area by Hofmann et al. [67] and Ebner and Strunz [71]. Recently this topic was covered by Triggs et al. [12] and Hartley and Zisserman [13].

Using camera pose (position and attitude) DEMs, orthorectified images, and other similar products can be produced through projections of the image rays. Like all measurements, camera pose measurements are corrupted by noise. Therefore, to improve the quality of these products, the camera pose estimates are often adjusted to enhance scene consistency using BA. BA uses only small set of corresponding features from overlapping images called tie points to improve the scene consistency.

BA for pushbroom imagery is more complicated than that for frame cameras, but is increasingly being used to refine pose parameters within the exploration community that are then fed into in semi-automated terrain reconstruction algorithms [15, 30], which may result in poor accuracy without indication. However, the adjustments made by BA for pushbroom imagery may have degrees of freedom (DOF) in the solution space beyond those commonly known from frame camera pose refinement. These DOF allow convergence to solutions that are inconsistent with the true camera pose. They depend on the geometric relationship

between the cameras pose used to obtain the imagery. Furthermore, the residuals minimized in BA do not provide any indication that an incorrect solution has been found. While several studies, for example [67], have investigated the practical constraints of BA with numerical techniques, there is no analytical discussion of this problem in the open literature. This work provides an analytical technique for studying this problem and uses it to analyze several geometries.

For orbital imagery of Earth, ground control points (GCPs) are commonly used to help constrain the BA process. An example is the use of GCP in the production of the recently released global DEM produced from ASTER data [72, 73] for Earth. However, the use of GCPs incurs significant expense. First precise position measurements need to be collected, either from GPS or existing maps. Then a human operator must manually identify the GCPs in the images. These requirements not only increase cost but also prevent full automation of terrain modeling. In addition, for certain terrains, GCPs are difficult to obtain e.g. arctic ice sheets [74] or extraterrestrial terrains [15].

To reduce manual processing time, ephemeris data (ED) generated from navigation system measurements on the spacecraft are increasingly being relied upon more heavily than GCPs. An example is SPOT5 satellite [75]. For many systems, navigational accuracy has been greatly increased by the Global Positioning System (GPS). However, in exploration, such as for Mars [30] or the Moon [76–78] this is not true, and GCPs are difficult to obtain and not accurate. Recent publications on DEM accuracy [74, 79] for SPOT5, which only makes use of GCPs periodically over calibration sites, report good accuracy. SPOT5 cameras were designed to be forward and aft looking to produce stereo imagery for terrain reconstruction, and similar camera configurations are being used for exploration orbiters [77, 78]. However, it is dangerous to assume that the multi line camera configuration of SPOT5 is responsible for its accuracy in DEM generation, and that this translates to the exploration orbiters. We will show that BA for the three line imaging geometry without either GCPs or ED has a degree of freedom not explicitly identified in other literature. Therefore, the

inaccurate GCPs and ED of in exploration missions may result in poor adjustment of the pose parameters.

A few authors make mention of DOF in BA, for example Triggs et al. briefly discuss the need to reduce the number of "gauge freedoms" [12] and several others refer to the commonly known scale and absolute position and orientation DOF without control points. A few other authors have performed simulation studies to determine the accuracy of terrain reconstruction for specific instruments under different conditions [80]. However, the literature is largely absent of any analytical discussion of the other DOF in pushbroom BA.

This chapter builds on BA formulation presented in Chapter 2 section 2.3. First we describe the problem that can occur when BA is used with images taken by pushbroom cameras. In many cases due to the missing along track perspective the Levenberg-Marquard minimization is not constrained and converges to erroneous solutions. Then we present an analytical approach that helps identifying scene geometries with deficiencies that lead to the described problems. Following are analysis that use developed method to evaluate constraints implicit to scene geometries commonly used to obtain satellite imagery. Another section illustrates the existence of the problem again, but this time by showing erroneous BA solution for the data obtained by Mars Reconnaissance Orbiter. Finally, analysis of simulation results for aerial imagery are presented. Chapter ends with short summary.

3.1 Problem statement

While a simple consideration of BA with pushbroom imagery might indicate that it can be constrained with a large number of tie points, there are many cases where this is not true. In the preceding development it is evident that BA for pushbroom imagery has more parameters than the same process for frame cameras, and therefore requires more tie-points to ensure more equations than unknowns. However, due to the lack of along track perspective, pushbroom imagery has a more insidious geometric problem. This section shows how selection of poor, and even commonly used, imaging geometry results in entire fami-

lies of equally consistent (minimal cost) terrain solutions with no way to distinguish which is correct. To illustrate this, we simulate the imaging of an artificial scene and perform sparse terrain reconstruction using tie points present in the BA. We compute the terrain using various initial pose estimates perturbed from that used to create the imagery. In each case, zero cost is achieved by BA, yet the camera pose estimates and reconstructed tie points differ significantly from the original. The terrain is not only shifted, but warped. It insidiously retains much of its original appearance, but any associated product such as a DEM or georeferenced image should not be used to accurately measure features. While it is well known that without control points stereo photogrammetry cannot recover scale, or position and orientation relative to an absolute reference even for frame cameras, the problem demonstrated here is different from this. The simulations illuminate the fact that products from pushbroom imagery may not even recover the terrain shape.

The simulated scene is depicted in Fig. 3.1. Images of the artificial terrain are created using a pushbroom camera model with both cameras traversing parallel trajectories in the X direction. The left image is nadir while the right image is the result of the camera looking to the side using only a rotation about the X-axis. Both images are constructed with the pushbroom array oriented perpendicular to the flight trajectory. Using the original pose parameters, the sparse terrain model (i.e., the tie points used in BA) can be reconstructed with very high precision, and the cost function is zero. The fundamental problem is that for this viewing geometry there exist a family of pose parameters that also have zero cost, but which have different terrain shapes. By holding one camera pose constant the perturbations and results constrain the well known freedoms to translate, rotate and scale the scenes. This illuminates other geometric DOF for this scene geometry.

To demonstrate such a DOF, the original images are used, but the right camera's attitude estimate is perturbed by adding small values to α_ω and β_ω, making ω a quadratic function of t rather than a constant as in the true attitude. All other pose parameters are unchanged. One might expect that for small perturbations BA could be used to recover the original pose.

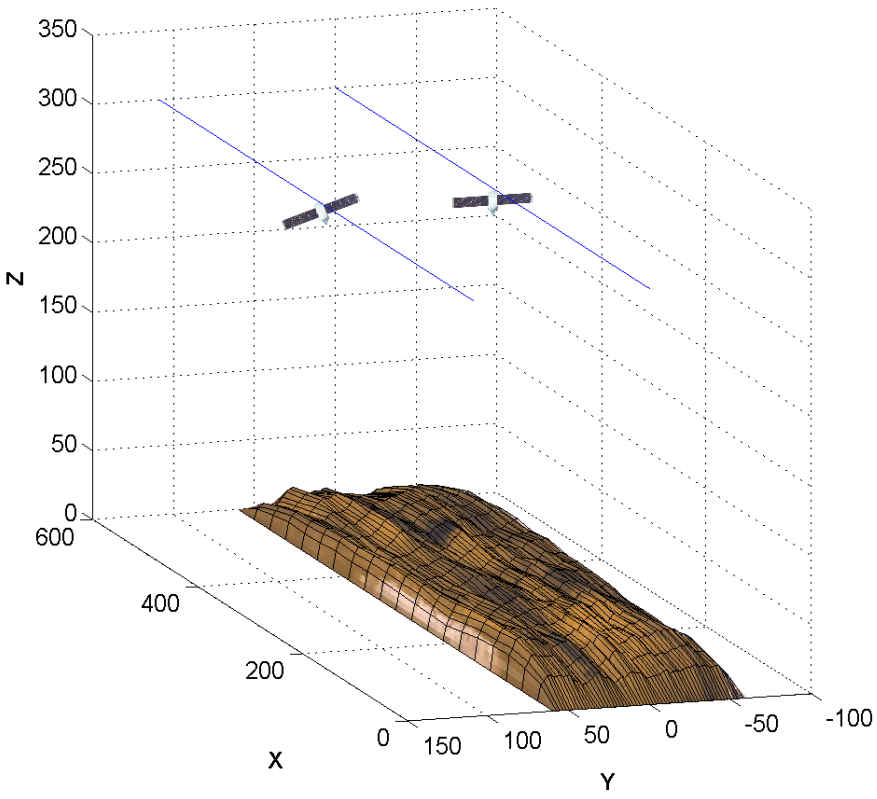


Figure 3.1: *Scene geometry illustration, the true terrain can be reconstructed. Note that the units are arbitrary, but scale between axes is one.*

However, BA's cost function is zero regardless of any perturbations in ω. The resulting perturbed terrain, again constructed from the tie points used in BA, is presented in Fig. 3.2. It is a warped version of the original. Small details are still visible and in general the distorted terrain resembles the original. However, the relative distances between points are not preserved. Since both configurations have zero cost, we are left with no estimate of the extent of the distortion.

This behavior is tied to the scene geometry. The cost function depends on the distance between corresponding rays emanating from the two cameras. When corresponding rays intersect, their contribution to the cost function C is zero. With the geometry depicted in Fig. 3.3, the plane formed by the pushbroom array and the focal point of the camera is

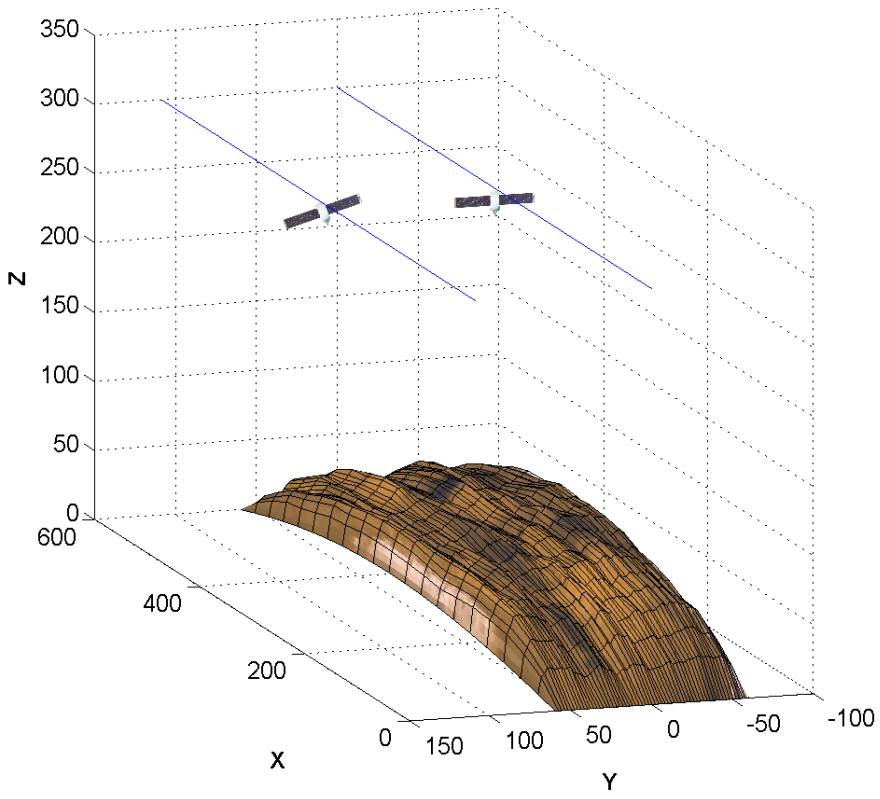


Figure 3.2: *Example of possible distortion of reconstructed terrain when pose parameters are disturbed, the cost C is numerically zero (in the order of 10^{-20}) but terrain is warped*

the same for both cameras at the times with corresponding features. Therefore the cameras are free to make any motion in this plane and the corresponding rays will still intersect. Specifically in this case the cameras are free to slide along Y and/or Z, and/or rotate around the X axis and all the corresponding rays for all features continue to intersect. Due to the lack of along track perspective, if the stereo imagery is achieved with only a rotation about flight trajectories that are parallel these planar DOF are still present, even if the flight trajectory is curved. The cost function, C for BA remains zero under any of these perturbations.

This particular geometric arrangement for camera trajectories, which is common, results in BA having six additional geometric DOF (three associated with each camera). This is

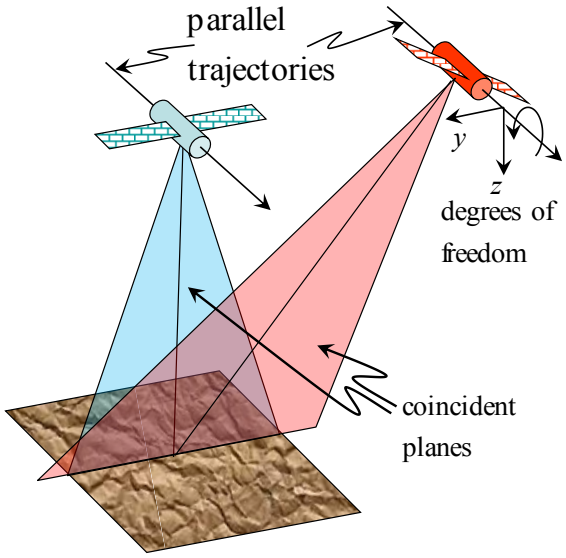


Figure 3.3: *Infinite number of possible points of intersection of two planes*

in addition to the seven DOF associated with translation, rotation and scaling of the entire scene found in frame camera BA. There exist a six-dimensional family of minimum cost solutions, each equally likely according to the cost function without control points. This is highly undesirable, as the BA process tends to converge to a solution close to a point in this family that is close to the initial estimate of pose without any indication of the resulting terrain's inaccuracy. Even if control points (ephemeris data and/or GCP) are used this is still undesirable because the control point measurements contain noise and because the DOF still exist between control points. It is desirable to use the tie points and control points together to offset the errors in all the measurements.

While the DOF in the example geometry used in this section may now seem somewhat intuitive, for the authors they were not so obvious until they were detailed with the following mathematical analysis. More importantly, other geometries contain DOF that are not so easily explained with a simple geometric analysis, and still other imaging geometries are more constrained and provide better terrain reconstruction. A mathematical method that helps identify the DOF for a specific geometry is developed in the next section.

3.2 Identifying the degrees of freedom

In this section, an approach is developed for identifying linear DOF in the BA process that do not change the minimal cost of a solution. First the problem is stated and analyzed mathematically. Then a process is described to perform the analysis needed to identify DOF for particular scene geometries.

We are interested in finding the set of directions, $\mathbf{s_i}$, in the parameter space along which we can move from the minimum without changing the cost. In other words we wish to find the directions that satisfy

$$C(\mathbf{k}_{min}) = C(\mathbf{k}_{min} + \epsilon \mathbf{s}_i). \tag{3.1}$$

for reasonably large ϵ. An ad hoc search for such a space is not feasible. We therefore need to narrow the possibilities for DOF using some analysis of the problem.

We will denote the gradient and Hessian of $C(\mathbf{k})$ at $\mathbf{k}_{min}$ respectively as $\mathbf{g}$ and $\mathbf{G}$. The curvature of $C(\mathbf{k}_{min})$ along the line $\mathbf{k}_{min} + \epsilon\mathbf{s}$ is then $\mathbf{s}^T\mathbf{G}\mathbf{s}$. It is well known [63] that a zero gradient and a non-negative curvature are necessary conditions for the existence of a minimum. Mathematically, these necessary conditions are

$$\mathbf{g} = \mathbf{0}, \tag{3.2}$$

and

$$\mathbf{s}^T\mathbf{G}\mathbf{s} \geq 0, \quad \forall \mathbf{s}. \tag{3.3}$$

Also it is well known [63] that a sufficient condition for the existence of a strict and isolated local minimum is

$$\mathbf{s}^T\mathbf{G}\mathbf{s} > 0, \quad \forall \mathbf{s}. \tag{3.4}$$

If we are at a minimum (i.e., Eq. 3.2 and Eq. 3.3 are satisfied) then the vectors satisfying Eq. 3.1 necessarily lie in the space not satisfying Eq. 3.4. The only difference between the set of vectors satisfying Eq. 3.3 and those satisfying Eq. 3.4 are those for which $\mathbf{s}^T\mathbf{G}\mathbf{s} = 0$. This is the null space of $\mathbf{G}$, for which an orthonormal basis is defined by the normalized

eigenvectors of $\mathbf{G}$ that correspond to zero eigenvalues. It does not matter if we speak of the left or right null space because $\mathbf{G}$ is symmetric assuming the 2nd partial derivatives are continuous.

If $\mathbf{G}$ is an $n \times n$ matrix having p eigenvalues that are zero, then it will have $n-p$ positive eigenvalues. An ordered set of eigenvalues for $\mathbf{G}$ is defined by $\zeta_n \geq \zeta_{n-1} \geq \ldots \zeta_{n-p} \geq \zeta_p = 0 \ldots \zeta_1 = 0$. An orthonormal basis for the row/column space and null space of $\mathbf{G}$ are defined using the normalized eigenvectors, respectively, by

$$\mathbf{S}_r = [\boldsymbol{\xi}_n \ldots \boldsymbol{\xi}_{n-p}] \tag{3.5}$$

and

$$\mathbf{S}_n = [\boldsymbol{\xi}_p \ldots \boldsymbol{\xi}_1] \tag{3.6}$$

Here $\boldsymbol{\xi}_i$ are the ordered eigenvectors corresponding to the ordered eigenvalues ζ_i. By using a diagonalization of $\mathbf{G}$ formed by $[\mathbf{S}_r \ \mathbf{S}_n]^T \mathbf{G} [\mathbf{S}_r \ \mathbf{S}_n]$ it easily shown than any vectors not satisfying Eq. 3.4 must lie in the null space defined by $\mathbf{S}_n$.

It should be noted that being in the null space of $\mathbf{G}$ is only a necessary condition for a vector to satisfy Eq. 3.1 when at a minimum. This is true because Eq. 3.4 is only a sufficient condition not a necessary and sufficient condition for an isolated minimum. Therefore the null space must be explored further to identify DOF. While it would be good to develop sufficient conditions for the cost to not change in a given direction, this is difficult. Consider the Taylor series expansion around $\mathbf{k}_{min}$ using an incremental displacement of $\epsilon \mathbf{s}$. This expansion is given by

$$C(\mathbf{k}_{min} + \epsilon \mathbf{s}) - C(\mathbf{k}_{min}) = \epsilon \mathbf{s}^T \mathbf{g} + \frac{1}{2} \epsilon^2 \mathbf{s}^T \mathbf{G} \mathbf{s} + H.O.T$$

where $H.O.T$ represent all Higher Order Terms. From this expansion we can see that in order for the cost to remain constant the sum of the second and higher order terms must be zero, implying the need to compute an infinite number of derivatives to find a sufficient condition.

For the sum of squares cost function, the problem of finding the Hessian $\mathbf{G}$ is significantly easier if the cost is zero. In general computing the Hessian involves finding the second partial derivatives of the cost function w.r.t. the parameters. However, if the cost is zero we only need to compute the first order derivatives associated with the Jacobian. It can be shown that for the sum of squares cost function the Hessian can be expressed as

$$\mathbf{G} = 2\mathbf{J}^T\mathbf{J} + 2\sum_{i=1}^{m} c(\mathbf{k})\nabla^2 c(\mathbf{k}). \tag{3.7}$$

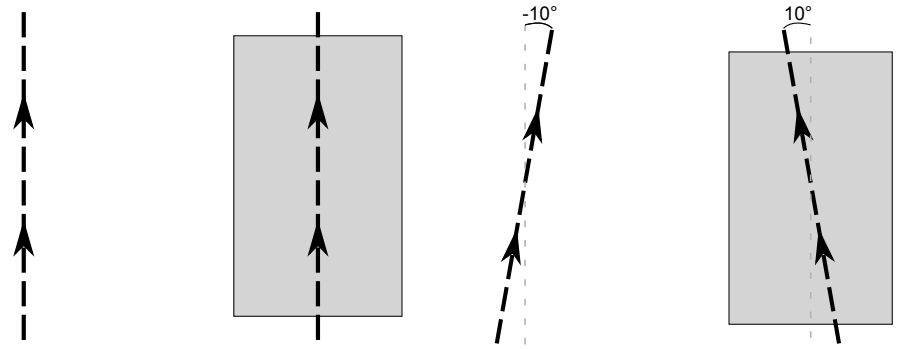
The second term of this equation is zero if the cost is zero.

The procedure we follow to search for DOF in a particular geometry is described by the following steps:

1. Form a simulated scene with true zero cost using the geometry.
2. Compute the Jacobian, and Hessian (using $\mathbf{G} = 2\mathbf{J}^T\mathbf{J}$).
3. Compute the zero eigenvalues of $\mathbf{G}$ and their corresponding normalized eigenvectors (e.g. using singular value decomposition (SVD)).
4. Test each of the directions given by these eigenvectors by moving along them from the zero cost solution while computing the cost. If the cost remains zero over a large range then consider it a degree of freedom and eliminate it from the space left to consider.
5. If the number of DOF identified is less than size of the null space minus one then it is still possible that a degree of freedom(s) lies in the set of vectors not eliminated from the search. Therefore, if possible, formulate a method to search this space in more detail.

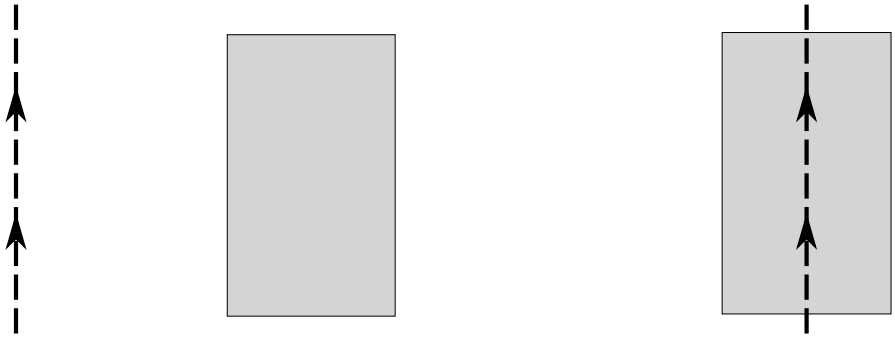
3.3 Analysis of viewing geometries

In this section results are presented from analyzing various scene geometries using the method described in the previous section. It is important to note that this analysis finds

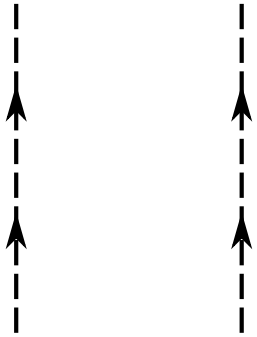


(a) Geometry 1

(b) Geometry 2



(c) Geometry 3



(d) Geometry 4

(e) Geometry 5

Figure 3.4: *Different scene geometries used in the analysis. Dashed lines with arrows represent cameras paths, gray rectangle represents terrain under observation.*

DOF in these geometries, it does not give relative measure of the conditioning of the scene. While it is true that properties such as base to height ratio and FOV are very important they are not the focus of this analysis. In fact, they have no impact on the DOF identified except in limiting conditions. What this paper emphasizes is that even for a scene that has very favorable base to height ratio and FOV, the existence of a DOF in the geometry can lead to erroneous pose solution. In this section, a reduction of the parameter space for BA is described, the geometries analyzed and computations are discussed, and the results of the analysis are given.

In this analysis it is desirable to formulate BA in such a way that the parameters are reduced to only those associated with the camera pose. This is desirable because the method described in the previous section involves the SVD of the square Hessian matrix for which the dimensions are defined by the size of the parameter vector $\mathbf{k}$. Furthermore, the DOF in BA associated with the camera pose are of particular interest because trajectories and attitudes can potentially be selected to constrain scene geometry and because they ultimately define the products produced with image projections. The feature locations are considered an implicit result of the camera pose. Using two ray intersection for the triangulation it is possible to eliminate the features from the parameter vector.

The parameter space is further reduced in this analysis by only allowing one of the two camera pose to vary. Fixing one of the camera paths fixes scale and fixes absolute position and orientation w.r.t. the reference coordinate system. It is well known, and intuitive, that BA without some type of external control such as GCP and/or ephemeris data can only solve for the relative position and orientation of the camera, and can only solve for the relative position with an unknown scale factor. Therefore allowing both cameras to move without external control results in at least the minimal seven DOF associated with identical motions of the two cameras and a scaling of all of the camera positions by the same value.

The parameter space is reduced still further by only allowing constant perturbations to the nominal camera path rather than the quadratic perturbations given in Eq. 2.28.

Allowing only constant path perturbations simplifies the analysis without loss of generality because adding higher order coefficients is redundant without external controls. This is true because any degree of freedom associated with a constant perturbation of a pose variable is also associated with the linear and quadratic perturbations of that variable. The parameter vector for the analysis performed in this section is therefore simply given by

$$\mathbf{k} = [\gamma_x \ \gamma_y \ \gamma_z \ \gamma_\omega \ \gamma_\phi \ \gamma_\kappa]^T \tag{3.8}$$

Five scene geometries are analyzed. They are presented in Fig. 3.4. Geometry 1 is that used for Fig. 3.1, Fig. 3.2, and Fig. 3.3. This is a common geometry for imagery from satellites with a single pushbroom sensor where two passes of a satellite are required to obtain a stereo-pair. One image is obtained from a nadir looking camera to obtain the best view of the area, while the other is taken from the side on a nearly parallel path by rolling the camera about the flight path for stereo triangulation. Geometry 2 presents a situation in which the trajectories from Geometry 1 were slightly rotated along vertical Z axis. This may be hard to realize in satellite imagery due to orbit constraints except near the poles in polar orbits. Geometry 3 and 4 illustrate cases when one of the cameras is offset in X and is fore looking and one or the other is offset in Y and looks to the side. Geometry 4 has an advantage of obtaining a nadir image. In Geometry 5 the cameras are fore and aft looking, which in terms of DOF is identical to a nadir view and either a fore or aft view. This geometry depicts another common way of obtaining stereo images from what have come to be known as "multi line cameras" (e.g. SPOT5 [75]). This geometry has the advantage of obtaining the image pair with very small temporal displacements on the same orbital pass. In all of these geometries the pushbroom sensor was oriented perpendicular to the camera path.

MATLAB$^{\text{TM}}$ software is used in all computations. In order to increase precision variable precision arithmetic (VPA) is used with 1000 decimal digits during Jacobian computation and the singular value decomposition of the Hessian matrix. This aids in distinguishing between eigenvalues that are truly zero and those that are small. To ensure accuracy

Table 3.1: *Degrees of freedom from eigenvector tests*

	Geometry				
	1	**2**	**3**	**4**	**5**
Nr of zero eigenvalues	3	2	2	2	2
Nr of freedom degrees	3	≤ 1 *	≤ 1 *	≤ 1 *	1

* concluded = 0 with subsequent analysis

the Jacobian is calculated using symbolically developed equations rather than a numerical method. These equations are very lengthy and cannot be presented here.

For each geometry a fictitious scene is generated. The set of 32 well dispersed tie points, chosen from the underlying terrain model, is projected into each of the two cameras CCDs using the true pose parameters, giving two artificial images of the terrain. Note that the cost function C for these true pose parameters is zero. The Jacobian for the reduced parameter vector in Eq. 3.8, the Hessian ($\mathbf{G} = \mathbf{J}^T\mathbf{J}$), and the SVD of the Hessian are calculated. Any eigenvector associated with an eigenvalue of zero is taken as a candidate for a DOF of the geometry. To evaluate the candidate DOF, $\boldsymbol{\xi}_i$, the camera pose is perturbed from the true camera pose, $[\mathbf{p}_0(t)^T \ \boldsymbol{\phi}_0(t)^T]^T$, in the direction of the candidate. In this case the camera EO is given by

$$\begin{bmatrix} \mathbf{p}_c(t) \\ \boldsymbol{\phi}_c(t) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0(t) \\ \boldsymbol{\phi}_0(t) \end{bmatrix} + \epsilon\boldsymbol{\xi}_i \tag{3.9}$$

where ϵ is a scale factor. This makes the cost a function of the scale factor, $C(\mathbf{k}) = C(\epsilon\boldsymbol{\xi}_i) = C(\epsilon)$. If the cost remains zero over a wide range of ϵ, then the candidate is taken as a DOF for the geometry. If all of these candidates are DOF then the number of DOF is clear, and so is their orthogonal set of directions. However, if there is more than one candidate that did not pass the degree of freedom test then it is still possible for DOF to lie in the space defined by these vectors.

Results for the five geometries are presented in Table. 3.1. For Geometry 1 three DOF were identified, namely along Y, Z, and ω, i.e., the non-fixed camera can be slid in Y (across-track), slid along Z (vertical) or rotated about the X axis without increasing the value of

the cost function. Sliding (rotation) can have any magnitude and sign, i.e., we can decrease the distance between cameras (rotation of a the camera) or increase it. This convenient and commonly used geometry is poorly constrained. For Geometry 5 a single degree of freedom, along X (along-track), was identified, therefore we can slide the non-fixed camera towards or away from the fixed camera without changing the cost function C. This along path direction is coincidently the most poorly known for ephemeris data from the navigation systems of many satellites. For each of the geometries 2, 3 and 4 two candidate directions, defining a two dimensional subspace, were identified that were combinations of the pose variables. In all three, after evaluation of $C(\epsilon)$ for a wide range of ϵ, neither candidate was found to define a DOF, leaving the possibility of isolated directions within the 2D subspace for which the costs may not change. However, further analysis below demonstrates that it can be concluded that these geometries do not have DOF beyond the well known scale and absolute reference DOF discussed at the beginning of this section.

For geometries 2, 3, and 4 it is feasible to study the 2D sub-space in more detail. To search for DOF in this space we evaluate the cost function in the plane defined by these vectors in circles around the known correct solution. In this case the camera pose is given by

$$\begin{bmatrix} \mathbf{p}_c(t) \\ \boldsymbol{\phi}_c(t) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0(t) \\ \boldsymbol{\phi}_0(t) \end{bmatrix} + \epsilon(\sin(\eta)\boldsymbol{\xi}_1 + (\cos(\eta)\boldsymbol{\xi}_2) \tag{3.10}$$

where η is the angle in this plane. This makes the cost a function of the angle and scale factor, $C(\mathbf{k}) = C(\eta, \epsilon)$. If the cost does not return to zero on the circle for reasonable values of ϵ, then it can be concluded that this space does not contain a degree of freedom for the geometry. A plot for geometries 2-5 of one such circular evaluation is shown in Fig. 3.5 using a value of $\epsilon = 10^{-4}$. Geometry 5 is included for comparison with a geometry with a degree of freedom known from the eigenvector tests. The degree of freedom for Geometry 5 is clearly identified in the plot. For all the other geometries the minimum cost is 6 orders of magnitude higher. We conclude that geometries 2-4 do not have DOF from the evaluation of several plots of this type where the minimum cost for Geometry 5 remains on the order

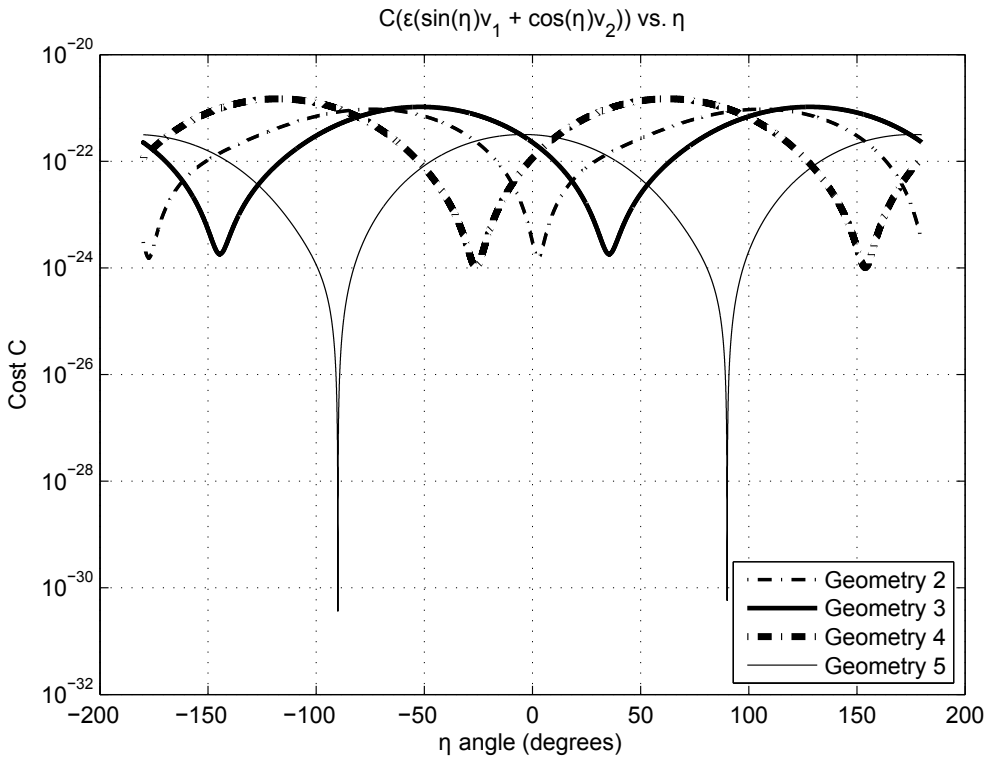
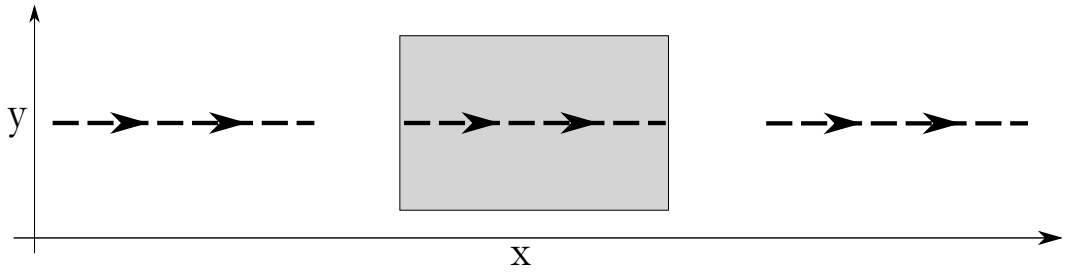


Figure 3.5: *Cost evaluated around the minimum on the circle lying in the plane defined by two eigenvectors* ξ_1, ξ_2 . $\epsilon = 10^{-4}$.

of that in Fig. 3.5 while the minimum cost for geometries 2-3 increase with increasing ϵ.

Similar analysis were performed in slightly different setup. This time positions of both cameras are fixed while their orientations are adjusted by BA. This scenario also eliminates six degrees of freedom, three for each of the cameras positions, but will isolate any degree of freedom associated with only rotation. Scale of the scene is set by the length of the paths and the distance between the paths. After rerunning the simulations, additional degrees of freedom were found again in both geometry 1 and geometry 5. For geometry 1 the degree of freedom was again found to be along ω . For geometry 5 the degree of freedom exist for simultaneous rotation along ω in the same direction of both cameras. In summary, three degrees of freedom were found for geometry 1 and two degrees of freedom were found for geometry 5. The other geometries are mathematically constrained (i.e., a single isolated minimum exists).



(a) Geometry 6

Figure 3.6: *Three line sensor geometry. Dashed lines with arrows represent cameras paths, gray rectangle represents terrain under observation.*

3.3.1 Three line camera case

In the analysis presented above two cameras were used in each geometry. However, Geometry 5 can be considered to be generated by the two separate CCD lines of a multi line camera. It is interesting to extend the analysis to the three line sensors commonly used for terrain extraction (e.g. MOMS-02 [81] or ADS40 [82, 83]). The usual configuration for a three line sensor can be modeled as one nadir looking camera, a fore looking camera, and an aft looking camera. This geometry, identical to Geometry 5 except for the presence of the nadir looking camera, is presented in Fig. 3.6 and is referred to as Geometry 6 from here on.

To use previous analysis techniques to determine existence or lack of DOF for Geometry 6 the testing procedure is modified slightly. First in order to eliminate the features, $\mathbf{p}_f$, once again from the parameter vector, $\mathbf{k}$, the triangulation method is extended to three ray case using multi-ray intersection. Using only two cameras for this triangulation would make tie points independent of the third camera's pose. Second, because fixing a single camera will eliminate the well known DOF associated with the scaling, rotation, and translation of the entire scene, the parameter vector must be augmented with the pose parameters of a second camera. We arbitrarily chose the nadir looking camera to be fixed. In this case Eq. 3.8 becomes

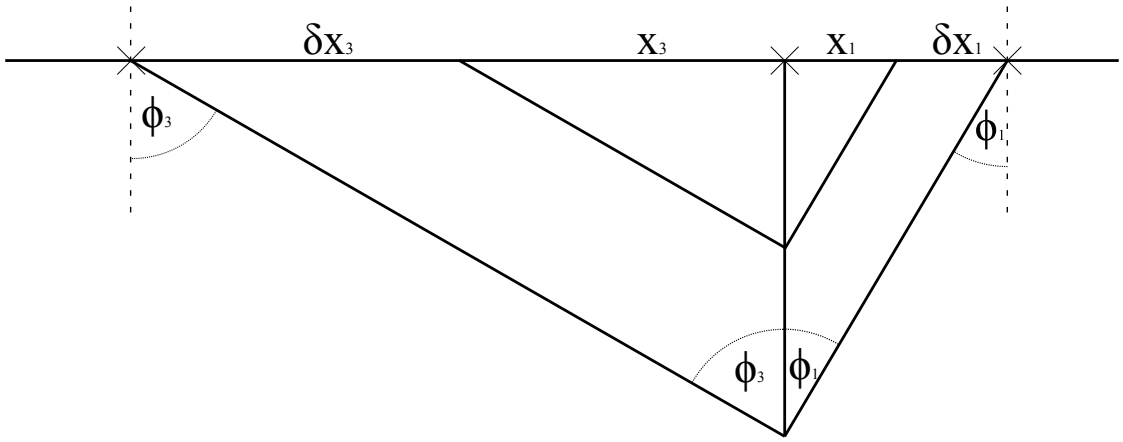


Figure 3.7: *Three line sensor geometry sliding resulting in no change in cost.*

$$\mathbf{k} = [\gamma_{x,1}\ \gamma_{y,1}\ \gamma_{z,1}\ \gamma_{\omega,1}\ \gamma_{\phi,1}\ \gamma_{\kappa,1} \\ \gamma_{x,3}\ \gamma_{y,3}\ \gamma_{z,3}\ \gamma_{\omega,3}\ \gamma_{\phi,3}\ \gamma_{\kappa,3}]^T \tag{3.11}$$

Again, an artificial scene is constructed, and 32 randomly dispersed tie points selected from the underlying terrain are projected into each of three images. The Jacobian is computed using developed analytical expressions. The Hessian is constructed, ($\mathbf{G} = \mathbf{J}^T\mathbf{J}$), and decomposed with the SVD.

The eigenvalues obtained by the analysis suggest a possible single DOF exists. The corresponding eigenvector coincides with sliding the fore and aft cameras along X (along-track direction) in a manner where one camera is proportionately slid in the opposite direction of the other camera. When the cost function is evaluated along this direction the existence of the DOF is confirmed. The ratio of the distances moved by the two cameras is determined by the angles of incidence (ϕ), as illustrated in Fig. 3.7. Assuming that the third camera slides a distance δx_3, the distance needed for the first camera to move to obtain zero cost is expressed as

$$\delta x_1 = (x_3 + \delta x_3)\frac{\tan(\phi_3)}{\tan(\phi_1)} - x_1$$

3.4 Example with Orbiter Data

To demonstrate the practical impact of the existence of DOF in BA, we illustrate their effect on the terrain reconstruction problem using images from NASA's HiRISE camera on MRO [84]. HiRISE data has been processed by several researchers using poor accuracy ephemeris data and/or GCPs (e.g. see Li et. al [30] and Kirk et. al [15]) to obtain DEMs. The accuracy of these DEMs should not be measured by the residuals of BA. We demonstrate here that although the BA process converges to a low cost solution, that other significantly different solutions exist that have similar cost.

A stereo pair of images capturing East Mareotis Tholus are used, *PSP_001364_2160* and *PSP_001760_2160*. Feature matching to find tie points and check points was performed with the SURF algorithm [60]. Over 20,000 features were successfully matched with sub-pixel accuracy. The RANSAC algorithm [85], using homography as an underlying model, was applied to eliminate rare mismatches. The SPICE kernel [86] provided the necessary information to translate pixel coordinates into focal plane coordinates and also corrected for optical distortion. An uniformly distributed subset of ephemeris data was chosen to be included in BA to provide control and avoid drift. For each image, BA was given a set of 400 observations of ephemeris data. Initial pose parameters were estimated by a polynomial fit the all ephemeris data available (approx. 20,000 points per image). With this data, pushbroom BA was performed with 400 tie points, withholding the rest to be used as check points for measuring the consistency of the results.

To check that the scene geometry was similar to Geometry 1, a simple test was performed. For each pair of image lines with a matched feature (tie point) normal vectors for the planes formed by each camera's CCD array and focal point were computed. The angle between these normal vectors is a measure of the coincidence of two planes. Computed statistics are given in Table. 3.2. They show that for this stereo pair the scene geometry resembles that described by Geometry 1 which has three freedom degrees.

A sparse terrain model model is given by the tie points after BA. To see if this terrain

Table 3.2: *Plane coincidence, angle (in expressed in degrees) between vectors normal to planes. Statistic for overlapping lines in both images.*

Mean	Min	Max	Std. dev.
1.3274	1.3242	1.3307	0.0017

Table 3.3: *Mean reprojection error (in pixels)*

		Non-disturbed	Disturbed
Before BA	u	17.92	208.11
	v	0.21	2.51
After BA	u	0.62	0.62
	v	0.01	0.01

model is unique, we perturbed the original path of one camera. Perturbation values were constant offsets from their measured values: 300m, 100m, 100m for $x_0(t)$, $y_0(t)$ and $z_0(t)$ respectively, 0.1mrad, 0.2mrad, 0.2mrad for $\omega_0(t)$, $\phi_0(t)$ and $\kappa_0(t)$ respectively. These perturbations are smaller in magnitude than the adjustments determined through BA by Kirk [15] for similar HiRISE data.

BA was performed again. The results from both consistency measurements are presented in Table. 3.3. In both cases the back-projection error was significantly decreased and the residual error for the check points have the same value. In both cases BA adjusted the path, and the tie points to be equally consistent with the HiRISE imagery. We will now show that the tie points represent a different shape.

As expected the sparse terrain models represented by these tie points are shifted and rotated from one another. Absolute position error of many hundreds of meters was found as the RMS of differences of the x_f, y_f, z_f of the same features in both point clouds, Table. 3.4. To measure the difference in shape we first found a transformation (translation and rotation)

Table 3.4: *RMS difference of two obtained terrain models (in meters)*

	Absolute	Shape
x	265.92	4.56
y	668.28	4.26
z	345.47	5.76

Table 3.5: *RMS of terrain error for 100 simulation runs for satellite imagery with assumed image noise of 0.5 pixel and path noise of* $[5m,\ 5m,\ 5m,\ 15e^{-6}rad,\ 15e^{-6}rad,\ 15e^{-6}rad]$

	Geometry									
	1		2		3		4		5	
	Init	BA	Init	BA	Init	BA	Init	BA	Init	BA
x	3.76	3.09	3.02	2.04	4.44	4.19	3.89	3.14	3.79	3.11
y	4.24	2.92	4.19	4.20	3.93	2.88	3.78	3.01	2.81	2.59
z	18.47	15.82	15.74	16.43	14.53	12.95	15.65	13.75	13.50	11.95
avg	**8.82**	**7.28**	**7.65**	**7.56**	**7.63**	**6.67**	**7.77**	**6.64**	**6.70**	**5.89**

that minimizes the sum of squared distances between respective points in both clouds. The residual of this sum after this transformation is then our measure of the difference in shape between the two terrains. This too is presented in Table. 3.4. The error in shape is less significant numerically than the shift, but may be more significant scientifically. One should be very careful when inferring some physical properties from DEMs generated in such a manner.

3.5 Simulations of Satellite Imagery

In this section computer simulation results performed for different scene configurations are presented. Computer simulations were performed to assess the influence of degrees of freedom on the accuracy of BA. A new simulation setup was constructed for this analysis. First a terrain model consisting of 512 points was created. Then, using one of the imaging geometries and a simple camera model resembling the QuickBird camera [87, 88], artificial images were generated. At this point, error-free terrain reconstruction was possible through triangulation. Next white Gaussian noise with a standard deviation of 0.5 pixels was added to the images to simulate quantization and feature detection errors in real images. Finally colored noise was used to disturb path measurements. The standard deviation of the noise added to path measurements was 5m for position and $15e-6$ rad for orientation. At this point, due to added noise, terrain reconstructed by the triangulations is significantly differ-

ent from the original terrain. Ten evenly distributed path measurements were selected and used as PCPs. These artificial noisy images and PCPs provide input to the BA algorithm. Then BA algorithm adjusted both camera poses and was allowed to run until convergence. RMS of the difference between BA reconstructed terrain and the original terrain defines the performance measure of the accuracy of BA. The above procedure was repeated 100 times and the average results are presented Table. 3.5.

In all cases the terrain reconstructed by BA is closer to the original terrain than the terrain obtained by initial triangulation and the accuracy gained through applying BA is similar for all geometries. However, there is no clearly visible advantage in using a mathematically constrained geometry over one that is constrained. This result appears to refute the previous analysis. To understand the discrepancy between the existence of degrees of freedom and BA's performance additional tests were performed.

Typical satellite imagery is from high altitude with a very narrow field of view (FOV). These high aspect ratios are an additional source of ill conditioning for BA which swamp out the geometric effects we address. On the other hand pushbroom imagery from air vehicles has significantly lower aspect rations due to both lower altitudes and wider FOV cameras. Examples of pushbroom cameras used in aerial missions include ADS40, ADS80 from Leica Geosystems [89, 90] or High Resolution Stereo Camera (HRSC) [91].

A second simulation was performed using a model of an aerial camera similar to HRSC with a FOV increased from 12 to 20 degrees. The focal length was 165mm and the pixels size was 6.5um and the flight altitude was set to 2000m. Path noise was set to 10m for the position and 5e-3 rad for the orientation. The standard deviation of image noise was set to 0.1 pixels. These values for the path distortion are large compared to the state of the art navigation systems [90]. However, such systems are prohibitively expensive for some users and lower cost system offer similar to worse accuracy compared to the one assumed.

The results for 100 runs comparing geometry 1 and 2 are presented in Table. 3.6. In this case, the degrees of freedom in the scene geometry do affect the accuracy of BA. The

Table 3.6: *RMS of terrain error for 100 simulation runs for aerial imagery with assumed image noise of 0.1 pixel and path noise of* [$10m$, $10m$, $10m$, $5e^{-3}rad$, $5e^{-3}rad$, $5e^{-3}rad$]

	RMS Error				Shape Error			
	Geo 1		Geo 2		Geo 1		Geo 2	
	Init	BA	Init	BA	Init	BA	Init	BA
x	6.08	5.85	6.08	5.64	0.24	0.21	0.28	0.01
y	6.95	5.53	6.99	3.99	2.01	1.56	2.04	0.05
z	19.35	15.19	19.54	4.51	0.11	0.09	0.27	0.03
avg	**10.79**	**8.86**	**10.87**	**4.71**	**0.79**	**0.62**	**0.86**	**0.03**

RMS Error is shown in the first two columns. The right two columns contain the shape error. Shape error is computed as the RMS difference between the original terrain and the recovered terrain once it has been shifted and rotated to best match with the original terrain. This measure is not influenced by simple translation or rotation but only by warping as it measures shape deformation.

Reconstructed terrain from the constrained geometry 2 is twice as accurate as the geometry 1 in terms of simple RMS error. In terms of shape error, geometry 2 provides twenty times better reconstruction than geometry 1. These simulations show that geometry can affect terrain reconstruction accuracy for aerial imagery. Moreover the developed method provides a tool for analyzing any geometry and provides insight into the nature of the degrees of freedom.

Even though, we were not able to demonstrate a practical advantage of one geometry over another with aspect ratio's typical for satellites, we were able to demonstrate the effects of geometry on the accuracy of terrain reconstruction under certain limited conditions. Another simulation, more similar to the analysis, with different perturbations on the parameters was performed. In this case, no image noise was added and only one camera pose was disturbed. The other camera's path parameters were left undisturbed and held fixed for BA. In this setup BA was always able to reconstruct the original terrain when geometries without degrees of freedom were used. However, the reconstructed terrain was always deformed with the unconstrained geometries, even though BA cost was reduced to 0. Reasonable values of

Table 3.7: *Terrain error as a function of a minimum BA cost.*

		BA stop condition - avg. cost per feature						
		1 pixel	0.1 pixel	0.01 pixel	$1e^{-3}$ pixel	$1e^{-4}$ pixel	$1e^{-5}$ pixel	0 pixel
Avg. terrain	Geo 1	0.239	0.239	0.235	0.235	0.236	0.219	0.223
error -	Geo 2	0.300	0.137	0.131	0.129	0.045	0.007	0.000
mean(x,y,z)	Geo 4	0.216	0.217	0.217	0.217	0.127	0.015	0.000

accuracy for localizing tie points impart so much noise at aspect rations typical to satellites that these errors swamp out the improvements gained using constrained geometry.

To explore the relationship between tie point localization noise and terrain accuracy the termination condition for BA was modified to approximate image noise. Under normal conditions BA runs until it converges (i.e., no better solution can be found). In this investigation, a weaker termination condition dependent upon the cost C was added. BA was stopped after it reached some predefined cost. Since the BA's cost is an aggregate image error, a weak termination condition is similar to feature localization noise. In this case, no image noise is added, but the pose parameters were perturbed. Table. 3.7 presents results of this simulation for geometry 1, 2 and 4. When the termination condition allows the cost to be relatively large BA exhibits no dependence on the scene constraints. As the termination condition is reduced, the performance for the constrained geometries improves. Unfortunately, to take advantage of the constraints of the scene, BA has to be able to converge to extremely small cost which on average is a fraction of the size of a pixel. Obtaining such a small cost is impossible with a real imagery, where features are identified with the accuracy up to one tenth of the pixel at best. This fact limits the practical significance of the identified degrees of freedom for the imagery obtained by the satellites.

3.6 Summary

In this chapter we described challenges that face BA when it is applied to imagery obtained with a pushbroom camera. A method for identifying degrees of freedom in viewing

geometries was developed. The method based on the decomposition of the Hessian matrix computed at the solution is able to identify linear directions along which the cost remains zero. The method was applied to six scene geometries that are commonly used to obtain imagery with pushbroom sensor. First five of those geometries assumed a single sensor on board of the satellite and image acquisition during two passes over a region. In two of these cases, certain degrees of freedom were found that should deleteriously affect terrain reconstruction accuracy. The last geometry under analysis assumed multiline sensor configuration that is also very common. A degree of freedom was found in this case as well. An experiment involving the real data captured by the satellite was performed to illustrate how harmful DOF are for BA. This experiment is aimed at cautioning the society about possible flaws of DEMs generated without proper control (e.g. in extraterrestrial applications). Numerical simulations under conditions resembling real camera systems were performed in order to verify expected gains in accuracy when using constrained geometries. It was found that for satellite imagery BA improves the terrain reconstruction similarly for all geometries, regardless of the presence of DOF in the geometry. Further analysis was able to attribute this discrepancy between theory and practice. For the satellite imagery the minimization problem is simply ill conditioned to start with and using constrained geometries does not eradicate this basic problem. However, for simulations of much better conditioned lower altitude aerial imagery, the scene geometry did affect terrain reconstruction accuracy particularly with FOV imagery.

Chapter 4

Parallel Tracking and Mapping for controlling VTOL airframe

4.1 Introduction

Improved accuracy, robustness, and capability of real-time processing in video-based simultaneous localization and mapping (SLAM) have recently made it a viable tool in a navigation solution for unmanned aerial vehicles (UAVs). However, obtaining an accurate and robust SLAM solution using only video has only recently been solved and not in a manner suitable for use on UAVs. These recent solutions require significant modification to be used robustly in the navigation of such a demanding application. But, the reward for a successful use of video based SLAM in these applications is high because vision based navigation can provide a very affordable technology, and because vision sensors have advantages over other sensors currently used in successful SLAM solutions for these applications.

Work presented by Klein and Murray gives a novel approach to vision based SLAM, parallel tracking and mapping (PTAM), wherein bundle adjustment (BA) is used instead of the typical filtering approach [42, 92]. The algorithm's accuracy and the robustness of their design are superb compared to any known real-time SLAM algorithm based on filtering. A key paper by Strasdat et al. compares performance of SLAM algorithms based on filtering and the SLAM algorithms based on bundle adjustment [43]. This work presents a modified version Castle's implementation of PTAM [93], parallel tracking and mulitple

mapping (PTAMM), for use in navigation on a vertical takeoff and landing (VTOL) UAV. PTAMM is available as an open source library. The PTAMM library works well for certain situations without modification, but is very limited in its application for UAV navigation without significant modification.

Very recent successes in using SLAM in real-time for navigation on small VTOL airframes have demonstrated its potential. However, nearly all of these successes have used SLAM solutions based on sensors other than pure vision. For example, several researchers have used LIDAR in their solutions including Bachrach et al.[40] Also, even more recently researchers have been successful in using depth cameras which return a dense array of pixels corresponding to the distance to the object seen by the pixel. However, both the depth cameras and the LIDAR which are small enough to be carried on a small UAV are very limited on range (on the order of 10m at best). Furthermore, current versions of the depth cameras cannot be used out doors, and both are significantly more expensive than simple cameras.

There are several examples of researchers working on video based SLAM for control of UAV. The authors are only aware of one other successful use of video-based SLAM in guidance of a UAV, other than that presented here. It is work performed by Bloesch et al. [37]. It shows that a PTAMM solution can be used to guide a UAV. However, in that work the UAV moves a very limited distance, with almost no rotation, in a scene where the map is generated prior to flight. It does not address the issues of building the map during flight. Also, Nuetzi et al. [39] present an approach to estimate a scale in PTAM by using data from an inertial measurement unit (IMU). But, this work presents only simulations and does not document any atcual flights. Other similar examples include researchers that collect data during flight and post process it off-line [34–36].

In this work the PTAMM algorithm is successfully modified to provide the position measurements necessary for the navigation solution on a VTOL UAV while simultaneously building the map. Furthermore, it is used in flights where large maps are constructed in real-

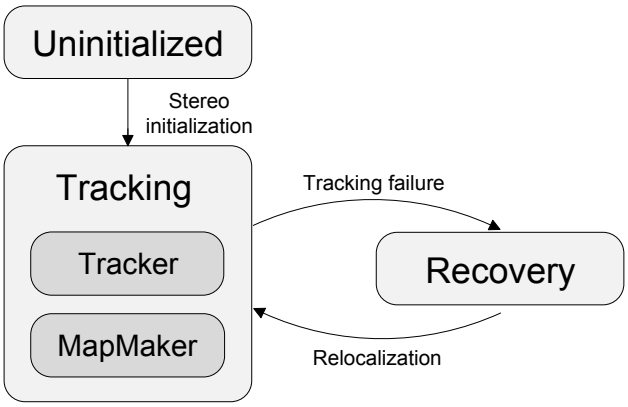


Figure 4.1: *PTAMM*

time as the UAV flies under position control with the only position measurements coming from PTAMM for the navigation solution.

4.1.1 PTAMM

This section gives an overview of the PTAMM algorithm. It gives a general overview of the entire algorithm, but only gives details in the parts that were modified to improve the algorithm's robustness in use for navigation with VTOL aircraft. For a more detailed description of the entire algorithm please refer to [9, 42, 92].

PTAMM builds a map of the environment by triangulating objects that are observed as matched features in images taken from significantly different vantage points. These images are called keyframes. At the same time it uses this map of 3D features and the current image to estimate the current camera pose by calculating it with the 3D features that are observed in this image. This is the normal operation when it is in the tracking state. However, PTAMM has three states:

1. Uninitialized
2. Tracking
3. Recovery

In the unitialized state, before the algorithm starts tracking, an initial map of 3D features is created using two frames. The user marks these two video frames which should be separated by a known distance and should observe the same part of the scene. This distance sets the scale of the entire map. These frames as stored in the algorithm as the first two keyframes and used throughout the tracking state. After marking a frame as the first keyframe FAST [62] is used to detect features in it. Those features are searched for in all the incoming frames. Searching uses correlation between patches of images centered around a feature. These patches are used to match it throughout the incoming frames. If the correlation is above a threshold the feature is declared found in the new frame. Otherwise it is excluded from further processing. After marking another frame as the second keyframe all the features that were successfully tracked from the first to the second keyframe are used to determine a homography between the two. The homography is decomposed to find the keyframes' poses by using the Faugeras and Lustman algorithm [94]. The distance between the camera positions is set to the user predefined value. BA is then used to refine the two camera poses and 3D feature locations. An epipolar search is then run to increase the size of map by finding more matches in the FAST features from the two keyframes. The algorithm moves to the tracking state.

Tracking is performed in parallel in two threads. The first thread, the Tracker, uses information stored in the map to find the camera pose for the current frame. The second thread, the MapMaker, maintains, extends and improves the accuracy of the information stored in the map. An overview of tasks performed by both threads is shown in Fig. 4.2.

The Tracker uses two stages, first the camera orientation is estimated by either a camera motion model or coarse image based minimization. Next, this orientation is used in the main tracking routine which estimates both the orientation and the position of the camera. In the default mode the motion model is not used, instead the Benhimane and Malis algorithm [95, 96] is used to roughly estimate camera orientation relative to the previous frame, and this is used for an initial guess in the main tracking algorithm. The main tracking algorithm

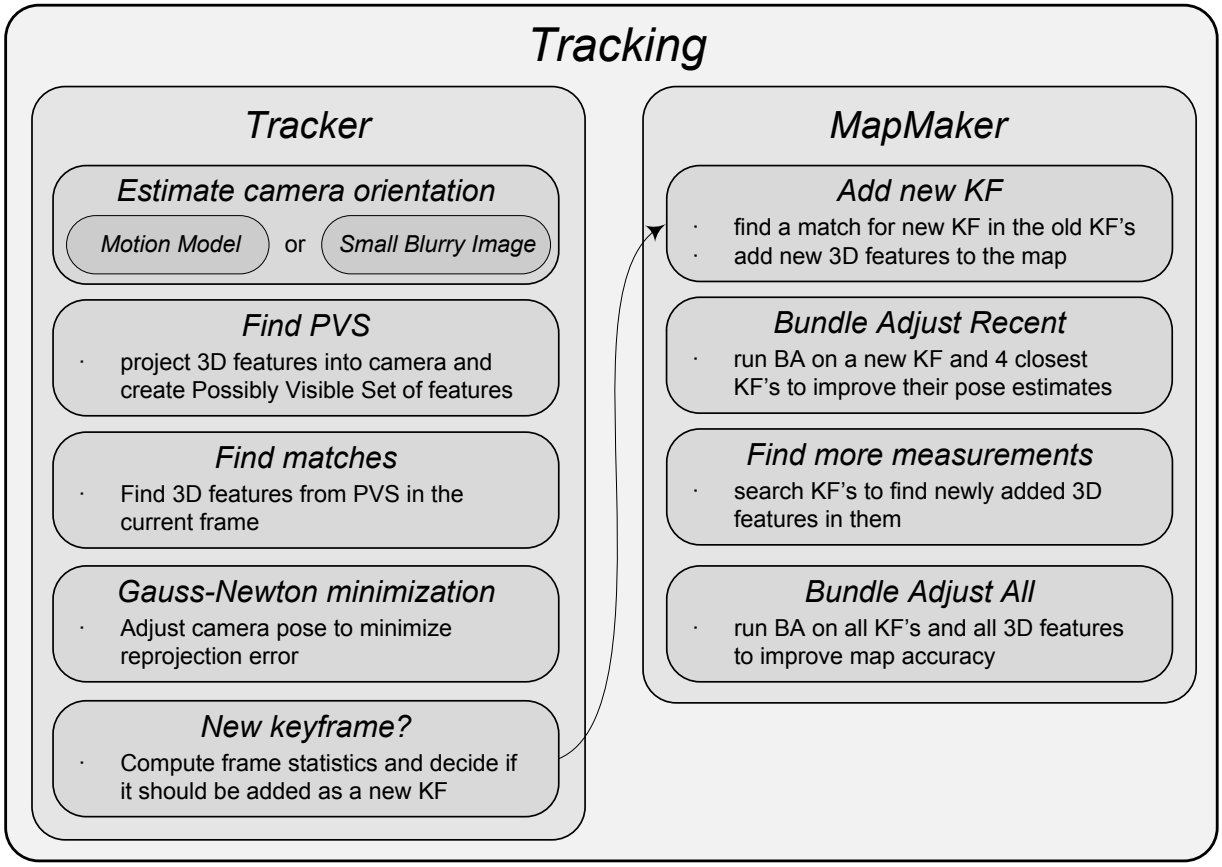


Figure 4.2: *PTAMM tracking overview*

is based on finding image patches associated with 3D features in the current video frame. For each feature, in the possibly visible set (PVS), a search is done locally around the location predicted by the projection of the 3D feature into the camera frame. This projection uses the initial guess of the current camera orientation computed at the first stage of the Tracker. To reduce the time for the feature search, only locations marked in the current frame as FAST features are considered. To help provide rotational invariance, feature patches are warped accordingly to the camera pose estimate. Once the image templates are matched a minimization process improves the estimate of the the camera pose associated with the current frame. This minimization tries to reduce the difference between actual feature locations and their predicted locations based on the current camera pose estimate.

The MapMaker only runs BA when new keyframes are added. Keyframes are added

using a normalized measure of distance to nearest keyframe. One key parameter calculated during tracking is the scene depth, which is the average of the distance to the features seen in the current frame. The normalized distance to the nearest keyframe is the distance to this keyframe divided by the scene depth. When the camera is moved to a new location that is further than a predetermined a normalized distance from any of the existing keyframes, the new frame is added as a keyframe.

The MapMaker maintains and extends the set of 3D features and the set of keyframes. When a new keyframe is added the Mapmaker uses an epipolar search between the new keyframe and one of the old keyframes to find new feature matches, which are triangulated to extend the map. BA is then performed with this new keyframe and and its four nearest neighbors to further refine poses and 3D feature locations. Then additional measurements of the new 3D features in other keyframes are added by searching for matches in their FAST features. Finally, BA is run with the entire set of keyframes and 3D features. Because this BA is much more computationally intensive than the Tracker algorithms, it is not capable of being at the typically required frame rates. Therefore the MapMaker is implemented in a separate thread that runs at much lower frequency than the Tracker.

PTAMM implements a recovery algorithm. Recovery is triggered when failure of the Tracker is detected. For each frame the Tracker computes the number of 3D features successfully identified in it. During normal operation most of the features in the PVS are found. However, significant changes between consecutive frames decrease number of features found because predicted feature locations are inaccurate due to the initial pose estimate being poor. Similarly, image blur decreases number of features found as a result of the features being smeared. Both of these can be caused by a fast camera motion and the significant inter-frame change can also be caused by limited frame rate. Without sufficient number of features or with enough false matches the Tracker minimization fails and the camera pose estimate drifts away from the true value. Since tracking of the current frame uses the pose estimate of the previous frame significant loss in tracking accuracy starts a chain effect that

usually leads to the failure of tracking. Once critical tracking conditions are detected, i.e., when the ratio of features found to the PVS in the frame drops below a threshold for a couple of consecutive frames, the algorithm stops relying on the pose information computed at the previous frames and PTAMM enters recovery. The current frame is compared to the stored keyframes, and the best matching keyframe is selected. Comparison uses zero-mean sum of squared differences and works on sub-scaled versions of the frames. 2D transformation of the selected keyframe into the current frame is computed using Benhimane and Malis algorithm [95, 96]. This computed transformation is used to estimate 3D camera rotation that would result in the computed image transformation. The Tracker is run with the initial position being equal to selected keyframe position and initial rotation being equal to computed 3D rotation. PTAMM recovers faster when the camera is pose is close to one of the stored keyframes poses.

4.2 Modifications to PTAMM for map building in navigation

In this section modifications to PTAMM are described that are made in an attempt to make it more robust in the application of navigation for VTOL airframes. First a modification to initialization part of the algorithm is described. Next changes to the keyframe addition and matching procedures are outlined.

4.2.1 SURF for initialization

During the initialization PTAMM uses a correlation based technique to track the features between frames as the camera moves from the first keyframe to the second. This approach is extremely error prone as any non smooth camera movement results in a rapid decrease in the number of features being tracked and forces a restart of the procedure. This prevents initialization in cases where the camera needs to travel significant distances in order to obtain reasonable stereo separation or when being transported by an aerial vehicle. Klein [97]

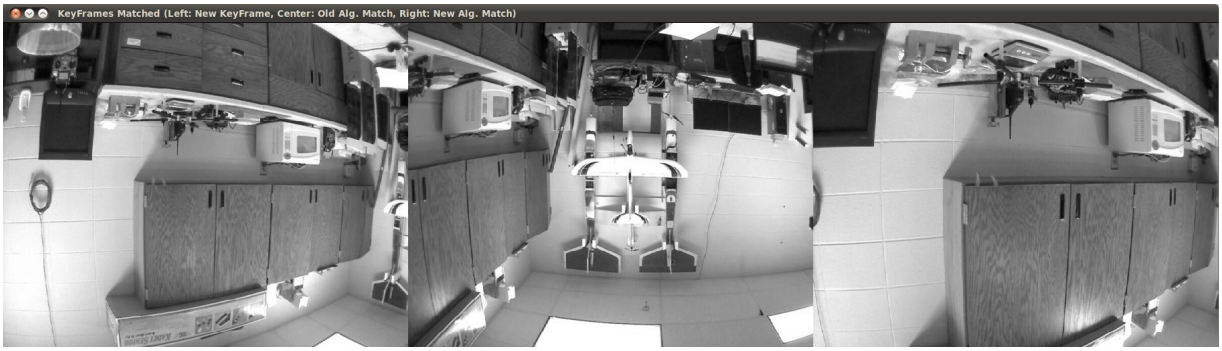


Figure 4.3: *Example of matching newly added keyframe. From left to right: newly added keyframe, PTAMM match for a new keyframe, proposed FAST based match. For the new method, note the overlap of the left parts of the keyframes which allows for addition of new 3D features to the scene. Using original PTAMM match would not initialize new features.*

identified this problem and proposed a two stage initialization. In the first stage tracking uses a single keyframe and operates using a homography. Features do not have 3D information attached and camera pose is not accurate. Once the camera is moved away from the first keyframe location and there is enough stereo separation between the views the algorithm switches into the normal mode.This approach allows initialization even when the camera movement is not smooth, however it assumes that initial scene is planar which is in general not true. Also, setting up the scene scale is less robust as the algorithm decides internally when to insert the second keyframe. If external devices like GPS or a barometric altimeter are used to set the scale their measurement may not be accurate or ready at this time.

We have chosen to implement an initialization procedure that uses the SURF [60] feature detector. After capturing the first keyframe SURF features are localized and the algorithm waits for the user to move the camera and mark another keyframe. No tracking is performed between the two keyframes, which allows the camera to be moved freely. When the user adds another keyframe SURF features are again found and matching is performed. The SURF implementation in the OpenCV library [98] is used to perform both feature extraction and efficient tree based matching. The rest of the initialization proceeds as in the original algorithm.

4.2.2 Fast map expansion

The algorithm for expanding the map in PTAMM works well for moving around an object while veiwing it from different locations or for moving in the direction the camera is looking. However it does not work well when new areas are viewed by rotating. In order to explore the environment (e.g. move in the corridor or go into a new room) user needs to carefully plan the pattern in which the the camera is moved. The map can not be expanded by simply pointing the camera into unknown part of the scene because a stereo view of any feature is needed to locate it. This poses challenges for exploration as two stereo separated frames picturing unmapped part of the scene are needed to expand the map with 3D features from this unknown part of the scene. PTAMM tries to expand the map only when a new keyframe is being added. Before adding a frame as a new keyframe the algorithm is required to determine:

- whether the current video frame contains new information useful for the algorithm? (part of the Tracker)
- and if yes, which old keyframe should be used for matching features with the new keyframe? (part of the MapMaker)

To answer the first question PTAMM uses normalized distance described in the previous section and to answer the second it uses the keyframe with the smallest linear distance from it. In neither case viewing angles are considered. This approach limits the algorithm's ability to explore the environment. Adding new features becomes difficult once an initial set of keyframes is captured because the camera location is likely to be close to at least one of the keyframes already in the set, although the current view may see a significantly different area due to rotation. This prevents fast and reliable exploration because areas without initialized 3D features stay unmapped. Finally, using closest keyframe for matching limits the possible stereo separation and the closest keyframe does not necessarily have the largest overlap of viewing area. We propose modifications changing keyframe handling that focus

the algorithm on expanding the map. As a result exploration is made easier.

A modified condition for adding a new keyframe is described here. The algorithm always adds a keyframe if the original, distance based criterion is fulfilled. If it is not, a new keyframe can be added based on the camera viewing direction. The viewing direction is compared with that of all keyframes within the threshold for normalized distance used to add keyframes. If the angle between current frame's viewing direction and a viewing direction of all keyframes in the described set is above a threshold we add the current frame as a new keyframe. The difference between viewing directions is just an angle between the two vectors representing them. This modification alone breaks the next part of keyframe addition - matching. The closest keyframe may not have significant stereo seperation and may not have significant overlap. Modification of the keyframe selection for matching is described next.

To ensure valid triangulation only keyframes having enough separation from the current frame should be considered for matching. To maximize the number of new 3D features added in the unmapped regions the keyframe selected by the algorithm should have a large overlap with the candidate frame in the region that is missing features.

Two approaches to accomplishing this were implemented and evaluated. In the first one the closest point of intersection of the camera viewing vectors is found to obtain a 3D point whose distance to the camera locations is compared to scene depths for the keyframes. The difference between expected point depth and the actual depth is used as a quality measure. A small difference suggests that the camera is looking at a similar area of the scene therefore a keyframe with the lowest difference is used for matching. In the second approach matching is run on scaled versions of the frames. This utilizes FAST features that were already found for tracking of the frame. A keyframe that has the highest number of matches with the candidate frame is selected for a full matching on the full resolution frames.

Practical evaluation of the two approaches reveals that while the first method gives instantaneous results, the second method finds the best keyframe to match with far more

often. The second method was chosen as the default implementation. Note that time requirements of the second method do not disturb tracking as keyframe addition is a part of MapMaker thread and it does not delay the Tracker. Fig. 4.3) shows a new keyframe and the keyframe chosen for matching using the original and new algorithms.

4.3 Modifications for handling low framerate video

Because our intended application of PTAMM might require use of significantly reduced computational power with an onboard processor the framerate becomes an important consideration. In this section we describe adaptations of the algorithm, aimed at allowing successful tracking even when changes in successive frames are significant. Although our initial implementation for navigation uses the results from offboard processing of the video with PTAMM, we intend to move this to an onboard processor. Reduced processing power results in the decrease of the number of frames that can be processed per second. This in turn leads to more significant differences between consecutive frames used by the Tracker. This phenomenon is further intensified when sudden accelerations are present (e.g. during wind gusts). Large image differences between consecutive frames is the usual cause for the Tracker to fail.

To estimate the camera orientation that is used to seed the Tracker PTAM uses what it refers to as small blurry images (SBI). The SBI algorithm uses the Benhimane and Malis algorithm [95, 96] which runs on 16 times sub-scaled and blurred versions of two frames. The Benhimane and Malis algorithm finds a 2D transformation (rotation and translation) converting the previous frame to the current frame. The returned 2D transformation is used in a minimization that finds a 3D rotation. The computed 3D rotation is applied to the pose of the previous frame to compute current frame pose.

PTAMM also includes a motion model for the initial pose estimate to replace the SBI algorithm. In our evaluation the SBI algorithm was found to behave much more reliably, especially when tracking at higher speeds or when direction of movement was changed

rapidly. Also, it is the default mode set by the PTAMM creators. However, when video frame rate is lowered and the camera motion is significant neither approach seeds the Tracker with a good orientation estimate and the Tracker fails. We describe two approaches that attempt to address this problem.

The first approach uses external measurements from sensors onboard the aircraft which measure angular rate. Most unmanned airframes are equipped with an inertial measurement unit (IMU) including accelerometers and gyroscopes that is used for flight stabilization. Gyroscopes provide angular rate measurements for rotation along the X, Y and Z axes of a body-fixed coordinate frame. The output of the gyroscopes are numerically integrated between camera frames using what is commonly known as the the "strapdown equation" in the navigation community. This provides an incremental update to orientation between frames, and is used to replace the SBI algorithm.

The second approach extends the SBI algorithm by seeding it with multiple starting points for the minimization. In the original implementation of the SBI algorithm the Benhimane and Malis algorithm is run once with an initial seed of zero rotation and zero translation. However, due to the low computational cost of this algorithm it is feasible to run the algorithm multiple times when processing a frame without incurring significant time delay. Therefore PTAMM was modified to detect failure of the Benhimane and Malis algorithm and turn on an "early recovery mode" where it is seeded with several initial values.

Failure is detected based on the final value of a cost function. The cost function is formulated as a difference between image intensities for the current frame and the previous frame transformed by the 2D transformation under estimation. Final cost is scaled by the inverse of the total number of pixels in the overlap area between the transformed and the candidate frame. If it exceeds a threshold a failure is assumed. The "early recovery mode" runs Benhimane and Malis algorithm starting with 27 different seeds of the 2D transformation. This transformation includes three variables: one rotation and two translations. The 27 different values are derived from all possible combinations of three values for each of the

variables. The rotation can take values of $(-45°, 0°, +45°)$, and the translations can take values of $-1/3 frames, 0 frames, +1/3 frames$. The solution with the lowest cost is chosen as base for a seed for the main minimization in the Tracker.

We also performed tests when the default single seed minimization is allowed 10 times more iterations than in the original PTAMM implementation. However test results (not shown in this paper) indicate that increasing number of iterations does not improve the results of the Tracker suggesting that the minimization must be converging to a non global minimum.

4.4 Cursory evaluation of tracking accuracy

In order to obtain a rough idea of the accuracy of the PTAMM solution a few experiments were performed in the scene shown in Fig. 4.6. In one experiment PTAMM was started and the map was initialized with the scale set to the precisely measured distance between initial keyframes. Then, the camera was moved along the lines of a rectangle drawn on the ground. The rectangle's sides measured $5m$ and $6m$. PTAMM tracking results were logged and Matlab was used to plot position estimates. As shown in Fig. 4.4, PTAMM reconstructed path closely resembles the rectangle. Recovered distances are within 5% of the expected ones.

In an another experiment, the camera was left stationary while PTAMM was allowed to track for 30 seconds. This experiment allows to estimate the position variance. Results are presented in Fig. 4.5, which shows that the variance is within $1cm$ in each axis. Both of the experiments were performed with the average distance to the features being around $20m$.

4.5 Evaluation of tracking performance

To evaluate changes to PTAMM two tracking experiments were performed. The first used a motion that had a single degree of freedom where a rotational motion stage was used to generate a known motion. However the parameters of the motion like the speed and

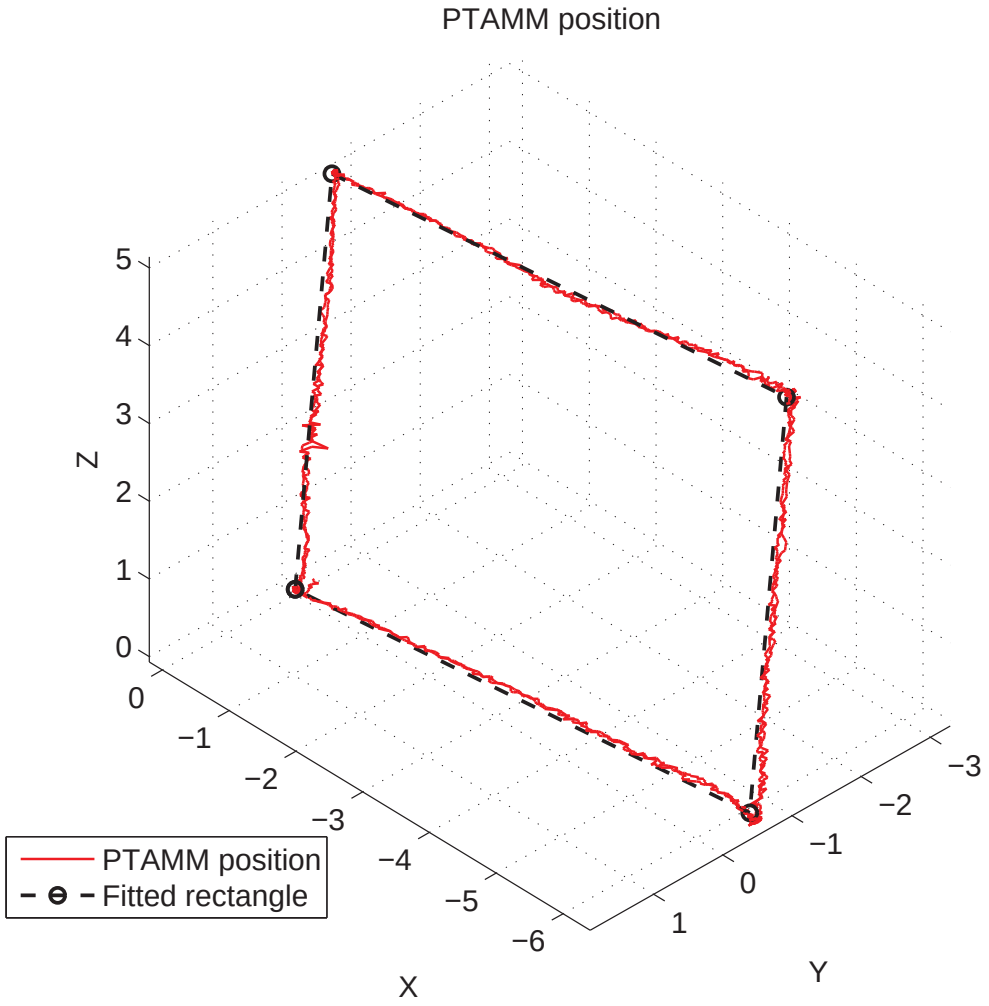


Figure 4.4: *Position computed by PTAMM for a rectangular movement. Actual PTAMM measurements are shown in solid line, fitted rectangle is shown in dashed line. For the actual movement the rectangle dimensions are* $5m$ *by* $6m$*. Rectangular fitted to PTAMM measurements is* $6.34m$ *by* 4.94 *which is within* 5.6% *and* 1.3% *of error in the first and the second dimension respectively.*

direction of movement were easily changed without changing other parameters. The second experiment used a freehand motion with 6 degrees of freedom. In both experiments all the data used by PTAMM were created prior to running the algorithm.

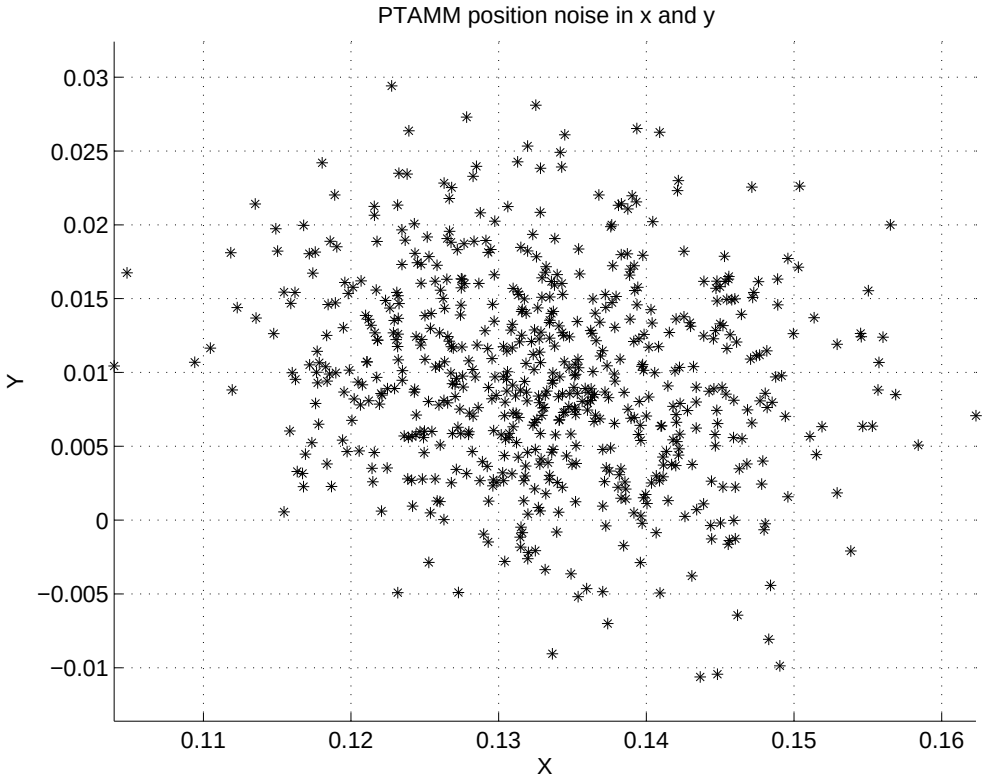
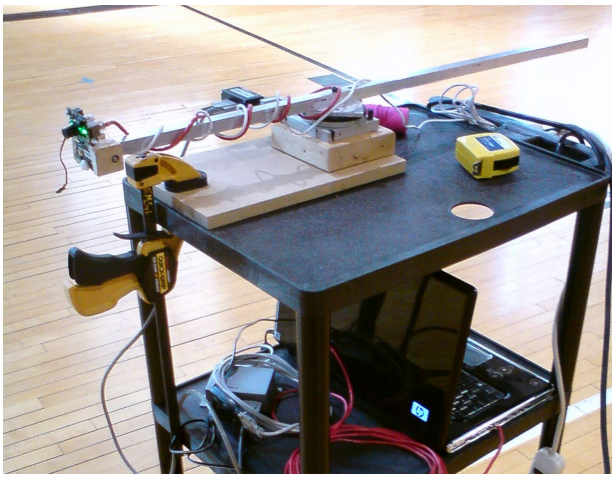
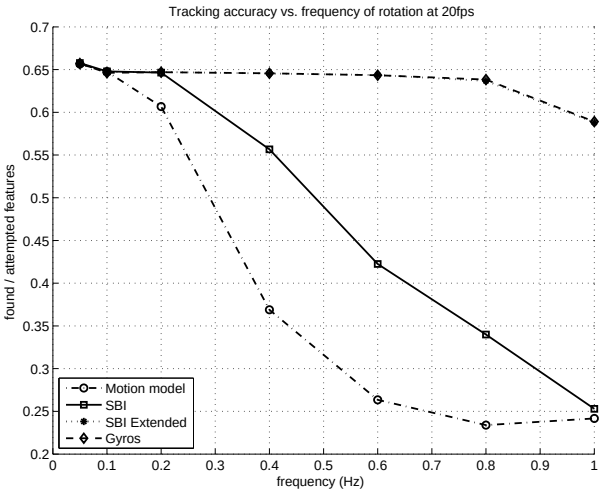


Figure 4.5: *Scatter plot of the x and y position estimates given by PTAMM for the stationary camera. Position estimate standard deviation estimated from this data is less than* $1cm$ *for each of* x*,* y *and* z*.*

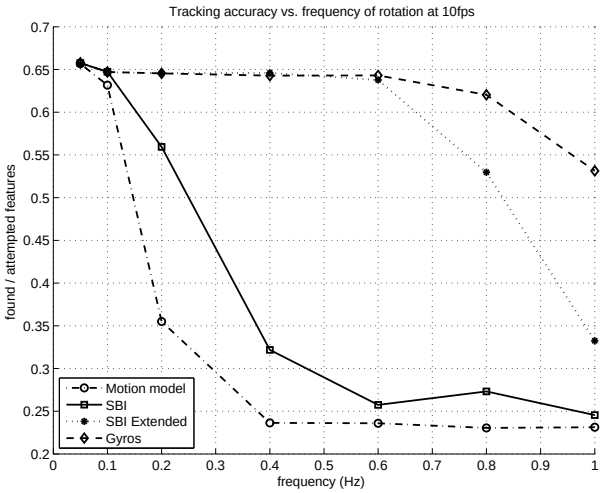
Figure 4.6: *Image created by stitching together some of the images used in the motion simulation. Stitched image covers* $160\,\mathrm{deg}$ *of rotation*



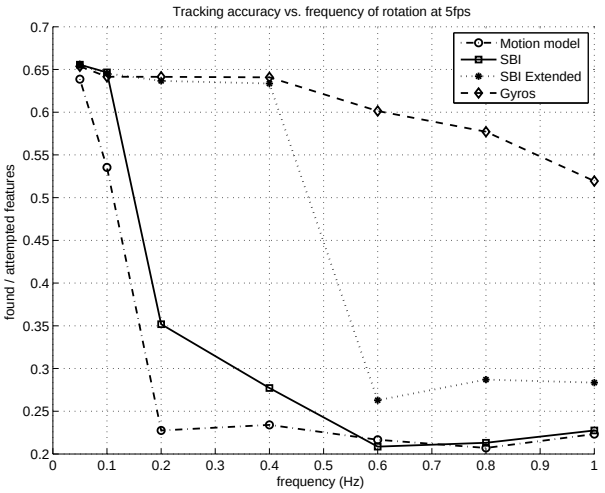
(a)



(b)



(c)



(d)

Figure 4.7: ***a*** *Setup used to capture data for motion simulation, Comparison of the tracking quality for different methods of estimating camera pose seeded to the Tracker and at different video framerates* ***b*** *20fps,* ***c*** *10fps,* ***d*** *5fps*

4.5.1 Stage Results

The camera was attached to a rotation stage. The stage offers a communication protocol that was used to query its orientation. Video data and stage position measurements were acquired and stored during a stage rotation encompassing 160 deg. In the resulting dataset each video frame has a corresponding measurement of the stage rotation assigned to it. This allows simulation of arbitrary motion in the single degree of freedom of the rotation stage. Rotations following a sine wave were used for the evaluation. The PTAMM video input was modified to use stored images. A frame rate limiting mechanism was also added. Additionally, PTAMM was run to create and save a map with features covering the entire area seen by the camera during the stage swath so that the same map was used for all experiments.

Evaluation of the method using gyroscope measurements $\boldsymbol{w}$ requires simulation of these measurements. Stage rotation measurements are used for that purpose. Due to camera mounting, stage rotation corresponds to rotation around Y axis in the camera frame. To simulate the measurements from the IMU an angular rate was approximated for Y axis using the difference in angular position measurements from the stage and the time from the simulation. Guassian white noise was added to this approximation, where the variance of the noise was obtained from data collected from the IMU on the airframe used in experiments discussed later.

A series of runs were performed using the described simulation setup to evaluate PTAMM behavior under different video frame rates. For each frame rate a sinusoidal rotation was simulated and appropriate video frames were fed into PTAMM. In all cases the amplitude of the motion was set to 80 deg however, the frequency of the sine wave was varied from $0.05Hz$ to $1Hz$. This corresponds to an average angular velocity being varied from around 2.5 deg /second to over 50 deg /second. Right before the start of the test saved map was loaded into PTAMM and locked to prevent any modifications. The motion sequence was started and PTAMM was allowed tracking for 30 seconds. Tracking results for each frame

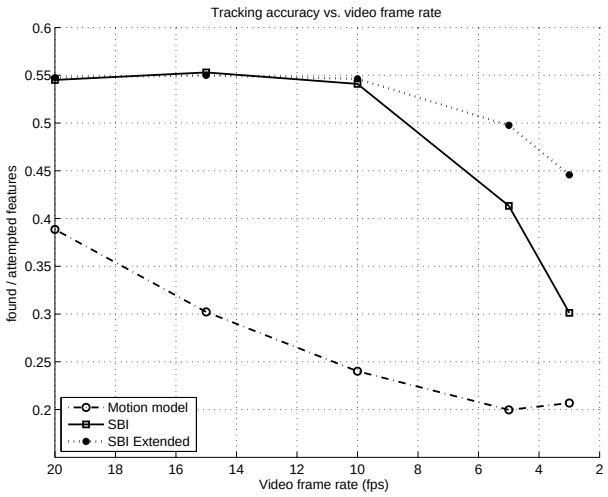


Figure 4.8: *Comparison of the tracking quality for a logged real date played at different frame rates*

were recorded and the ratio of the number of features found in the frame to the number of features in the PVS was used as a performance metric.

Four algorithms were compared. Two standard PTAMM approaches: motion model and the SBI algorithm, and two proposed algorithms: the extended SBI algorithm and use of gyroscope measurements. The results of experiments are presented in Fig. 4.7. In all cases the modified methods improve the ability of the Tracker to estimate camera pose at higher motion speeds. Improvement is more clear at lower frame rates. The method using gyroscopes gains advantage over extended SBI at low frame rates and at high motion frequencies due to decreased overlap between frames caused by the motion.

4.5.2 IMU results

Another experiment used data captured during the freehand camera motion. Video input and gyroscopes measurements provided by the IMU attached to the camera were timestamped and saved at the PC. A PTAMM map of the scene was created before running the tests and stored as before. The captured video sequence was input to PTAMM at various frame rates.

Results are presented in Fig. 4.8. Extended SBI performs better that the default version

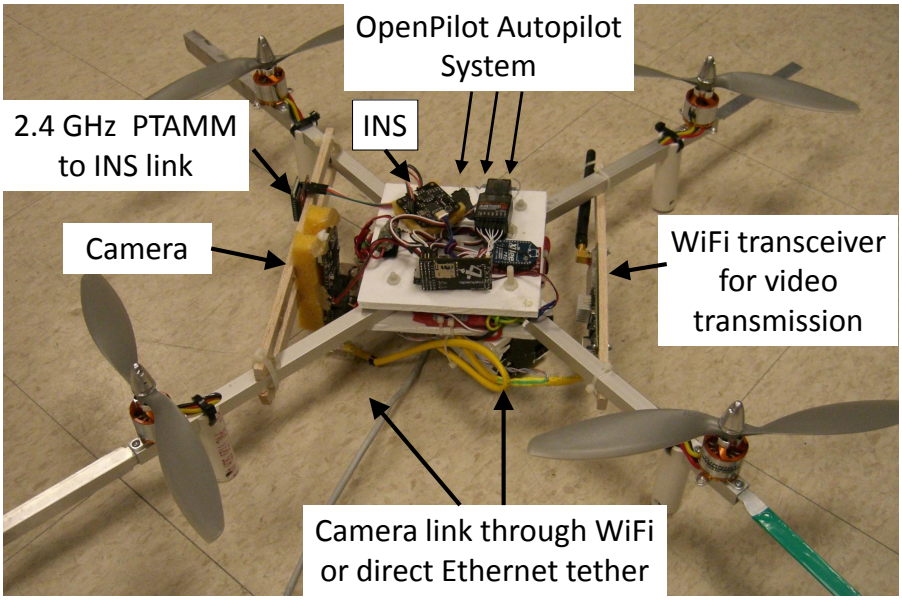


Figure 4.9: *Multi-rotor airframe modified for PTAMM*

at the lower frame rates, however the difference is as significant as in the previous results. Gyroscope based tracking performs very poorly though and is not shown in the figure because it failed in all cases. This might be the result of a couple of factors. Different time delays encountered on communication links from camera and from IMU can create a time offset between the data from each. An inaccurate estimate of the constant rotation IMU and camera may have caused too large of errors.

4.6 PTAMM airframe integration

Integration of PTAMM with the airframe used for testing involved both hardware and software modifications to multi-rotor UAV. An annotated picture of the airframe is shown in Fig. 4.9. In the current implementation, the PTAMM solution runs on a laptop computer in the Linux operating system, rather than on board the aircraft. Therefore the images are transferred to the ground either directly through an Ethernet cable or through an 802.11n

wireless transceiver. The camera is AXIS M1054, an IP security camera, with the resolution set at 800x500 pixels. The wireless transceiver is capable of transmitting well over 30 frames per second with this resolution, which is not a limitation because the PTAMM algorithm on the laptop processes approximately 15 to 20 frames per second in normal operation.

The PTAMM algorithm communicates with the OpenPilot autopilot through a 2.4 GHz serial communications transceiver connected to the INS board of the autopilot. The OpenPilot system is an open source autopilot with hardware consisting mainly of an INS board, a main control board, a GPS receiver, and a 2.4 GHz serial communications link with the ground station. The INS board contains the IMU sensors and a microcontroller, which implements the navigation solution. The authors have contributed to several parts of the development of OpenPilot and most significantly to the development of the navigation solution.

The navigation solution in OpenPilot is an EKF implementation of an INS. It takes 3-axis accelerometer measurements and 3-axis rate gyro measurements as the inputs to the dynamic system modeled in the EFK. In addition, it uses a 3-axis magnetometer, GPS position, and GPS velocity as the measurements of the outputs of the system. The dynamic model used is a 6DOF kinematic model of a rigid body. When using PTAMM with the navigation solution the GPS is not used. Rather, the position measurements from PTAMM replace the GPS position measurements and no velocity measurements are used. Also, when using PTAMM the magnetometers are not used, so the yaw angle of the airframe is not observable without another measurement. Since the both PTAMM and the INS represent orientation with a quaternion the simplest solution to this is to add an additional output/measurement to the INS which corresponds to the forth element of the quaternion. This element of the quaternion is highly correlated with yaw.

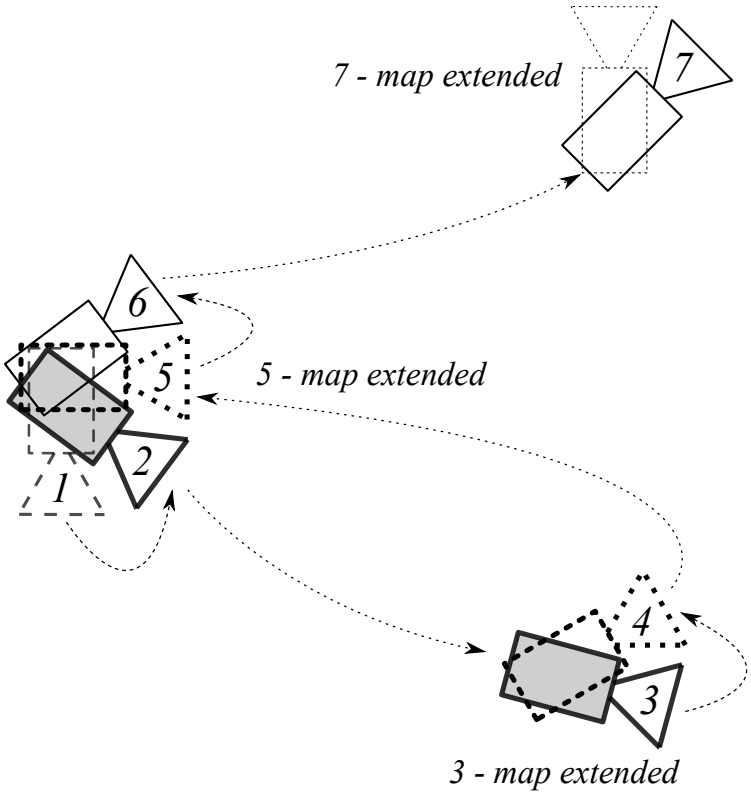


Figure 4.10: *Map building with the new keframe addition. Camera successive positions where new keyframe is added are numbered from 1 to 7. 1 - PTAMM after stereo initialization, 2 - camera rotated to point into part of the scene without initialized features, 3 - camera moved forward to obtain stereo separation, keyframe added and new features initialized, 4 - camera rotated again to look at another uninitialized part of the scene, 5 - camera moved back for stereo, new features added, etc.*

4.7 Test Flights

The exploration and navigation capabilities of the modified PTAMM algorithm were verified in test flights. As shown in Fig. 4.9 the camera looks horizontally as it is mounted on the UAV. Therefore rotating the UAV around the Z axis (yaw) point it towards new parts of the scene. To add to the map features in a new area PTAMM needs two images(frames) of that area separated in space to allow for a stereo view. It is possible to generate large maps during flights under stable control using the navigation solution being generated with position and yaw measurements from PTAMM. This is done fairly easily with the modifications, but would be very difficult with original algorithm. A possible routine for moving and rotating

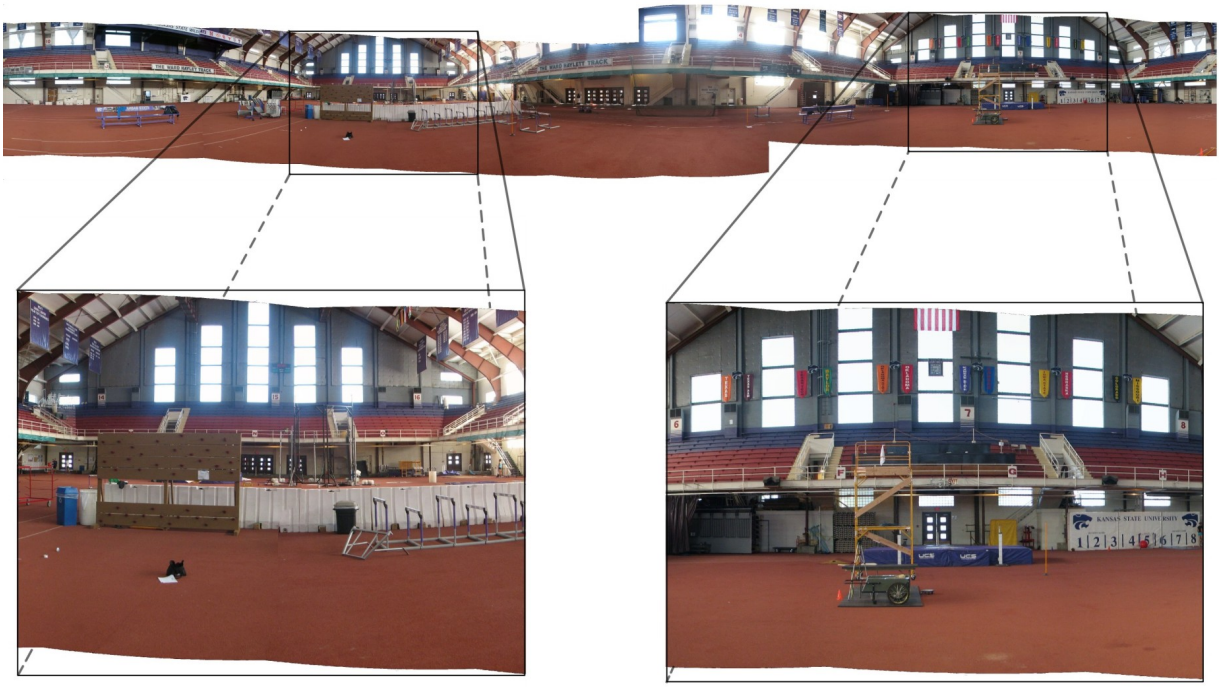


Figure 4.11: *Panoramic view of a field house. One end of the field house is approximately 25 meters from the camera, other is around 125 meters from the camera.*

the camera to extend the map is presented in Fig. 4.10. The map is created by moving in and out in a "circle", with multiple keyframes being added at the center and two keyframes being added at the rim of the "circle".

Fig. 4.11 shows a panoramic view of a 360 degree scene in a field house. It was possible to generate a map of nearly the entire 360 degree field of view very simply during flight controlled very stably by the navigation solution. This was done by rotating and moving back and forth a distance of about four meters. The two extreme ends of the field house, which are broken out in this figure, were approximately 25 meters and 125 meters away from the flight location. Despite the small stereo separation the only portion of the map where the UAV developed poor stability was when it was facing the end of the field house the furthest away. In this area nearly all of the features identified were very far away in the upper reaches of the field house and the PTAMM solution showed significant noise in the position solution.

Holding the UAV in hand an entire 360 deg map was generated in the same way. In this case however the map was closed through the full rotation. The UAV was then subsequently flown in this map even in the area where it was closed by rotation. Again the only problem area was when the camera was facing the furthest end of the field house. It was possible for the navigation solution to hold the UAV in one position while simply rotating 180 deg with this prebuilt map.

4.8 Summary

This chapter presented a vision based system for UAV navigation. Modifications to a leading SLAM algorithm, PTAMM, were developed, described and verified. After modifications, the algorithm is suitable to work with the UAV navigation system and shows better exploration capabilities. PTAMM extensions for handling low frame rate video were also proposed. Integration of external sensors into PTAMM was investigated, but the comparison to the existing implementation was inconclusive. Simulations show a promise of improved robustness which is not confirmed by the tests using hardware. However, extension to the coarse Tracker routine showed consistently better performance than the original implementation. The last sections of this chapter described the hardware used, including the four rotor airframe, camera, and wireless communication system, and presented results of the test flights where the UAV was controlled with use of PTAMM tracking results.

Chapter 5

Conclusions

The main interest of this dissertation is localization of objects and mapping of the environment. First, the problem of improving estimates of the camera pose in aerial and satellite imagery using imagery was studied. In particular, imagery obtained with pushbroom cameras was of interest. Contributions of this work include: 1) formulation of a method for detecting DOF in the BA; and 2) identifying that two camera geometries commonly used to obtain stereo imagery have DOF. This implies that there is no isolated global minimum that BA can converge to without other external controls. It is hoped that this observation brings much needed caution to the scientific community involved in extraterrestrial research that is relying on DEMs produced using imagery acquired with pushbroom cameras. Finally, the work presents results demonstrating that avoidance of the DOF can give significant accuracy gains in aerial imagery.

The second part of this dissertation describes an approach for developing a monocular vision based navigation system for UAVs. This system is built by modifying a leading SLAM algorithm to address challenges created by its use in real time navigation for a VTOL airframe. The map building capabilities are significantly improved and the robustness under low frame rate video is improved. The SLAM solution is successfully modified to provide the position measurements necessary for the navigation solution on a VTOL UAV while simultaneously building the map. It was used in flights where large maps were constructed in real-time as the UAV flew under position control with the only position measurements for

the navigation solution coming from SLAM. The main contributions of this part include: 1) extension of the map building algorithm to enable it to be used realistically while controlling a VTOL UAV and simultaneously building the map; 2) improved performance for low frame rates; and 3) the first known demonstration of a monocular SLAM algorithm successfully controlling a UAV while simultaneously building the map. Practical application of this solution still needs significant effort in order to eliminate some of the discovered problems. However, this work demonstrates that a fully autonomous UAV that uses monocular vision for navigation is feasible.

Future work on the monocular SLAM algorithm for control of VTOL UAV is needed to make it robust and more useful in such applications. This work would likely include two major things: 1) moving it to an on board processor; 2) tighter integration with the control system to control the motion of the UAV while extending map to avoid loss of tracking.

Bibliography

[1] M. J. Broxton, R. A. Beyer, Z. Moratto, M. Lundy, and K. Husmann, *The Ames Stereo Pipeline*, NASA Ames Research Center, http://ti.arc.nasa.gov/tech/asr/intelligent-robotics/ngt/stereo.

[2] F. F. Sabins, *Remote Sensing Principles and Interpretation*, 2nd ed. W. H. Freeman, 1978.

[3] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *Robotics Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99 –110, 2006.

[4] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping: part II," *Robotics Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108 –117, 2006.

[5] A. Kleiner, E. Kleiner, J. Prediger, and B. Nebel, "RFID technology-based exploration and slam for search and rescue," in *In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS*, 2006, pp. 4054–4059.

[6] J. D. T. Pedro Piniés and J. Neira, "Localization of avalanche victims using robocentric SLAM," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, China, October 2006.

[7] Z. Zhang, H. Guo, G. Nejat, and P. Huang, "Finding disaster victims: A sensory system for robot-assisted 3D mapping of urban search and rescue environments," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 3889 –3894.

[8] D. Sun, A. Kleiner, and T. M. Wendt, *Multi-robot Range-Only SLAM by Active Sensor Nodes for Urban Search and Rescue*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 318–330.

[9] R. O. Castle and D. W. Murray, "Object recognition and localization while tracking and mapping," in *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality*. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 179–180.

[10] R. O. Castle, G. Klein, and D. W. Murray, "Combining monoSLAM with object recognition for scene augmentation using a wearable camera," *submitted to Image and Vision Computing*, vol. 28, no. 11, pp. 1548 – 1556, 2010.

[11] C. McGlone, E. Mikhail, and J. Bethel, *Manual of Photogrammetry*, 5th ed. American Society of Photogrammetry and Remote Sensing, 2004.

[12] B. Triggs, P. F. Mclauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice*. Springer Berlin / Heidelberg, 2000, pp. 298–372.

[13] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[14] A. S. McEwen, E. M. Eliason, J. W. Bergstrom, N. T. Bridges, C. J. Hansen, W. A. Delamere, J. A. Grant, V. C. Gulick, K. E. Herkenhoff, L. Keszthelyi, R. L. Kirk, M. T. Mellon, S. W. Squyres, N. Thomas, and C. M. Weitz, "Mars Reconnaissance Orbiter's High Resolution Imaging Science Experiment (HiRISE)," *J. Geophys. Res.*, vol. 112, 2007.

[15] R. L. Kirk, E. Howington-Kraus, M. R. Rosiek, D. Cook, J. Anderson, K. Becker, B. A. Archinal, L. Keszthelyi, R. King, A. S. McEwen, and H. Team, "Ultrahigh resolution topographic mapping of Mars with HiRISE stereo images: Methods and first results," in *Seventh International Conference on Mars*, 2007, pp. 1428–1429.

[16] G. Chin, S. Brylow, M. Foote, J. Garvin, J. Kasper, J. Keller, M. Litvak, I. Mitrofanov,

D. Paige, K. Raney, and et al., "Lunar reconnaissance orbiter overview: The instrument suite and mission," *Space Science Reviews*, vol. 129, no. 4, pp. 391–419, 2007.

[17] M. S. Robinson, S. M. Brylow, M. Tschimmel, D. Humm, S. J. Lawrence, P. C. Thomas, B. W. Denevi, E. Bowman-Cisneros, J. Zerr, M. A. Ravine, and et al., "Lunar Reconnaissance Orbiter Camera (LROC) instrument overview," *Space Science Reviews*, vol. 150, no. 1-4, pp. 81–124, 2010.

[18] D. C. Brown, "A solution to the general problem of multi station analytical stereotriangulation," Patric Airforce Base, Florida, Tech. Rep., 1958.

[19] H. Schmid, "Eine allgemeine analytische Lösung für die Aufgabe der Photogrammetrie," *Bildmessung und Luftbildwesen*, vol. 4, pp. 103–112, 1958.

[20] E. M. Mikhail, J. S. Bethel, and J. C. McGlone, *Introduction to modern photogrammetry.* Wiley, 2001.

[21] H. M. Karara, *Non-Topographic Photogrammetry*, 2nd ed. American Society of Photogrammetry and Remote Sensing, 1989.

[22] K. B. Atkinson, Ed., *Close Range Photogrammetry and Machine Vision.* Caithness, Scotland: Whittles Publishing, 1996.

[23] M. A. Lourakis and A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment," *ACM Trans. Math. Software*, vol. 36, no. 1, pp. 1–30, 2009.

[24] M. I. A. Lourakis and A. A. Argyros, "Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?" in *Proceedings of the Tenth IEEE International Conference on Computer Vision.* IEEE Computer Society, 2005, pp. 1526–1531.

[25] R. Li, G. Zhou, N. Schmidt, C. Fowler, and G. Tuell, "Photogrammetric processing of high-resolution airborne and satellite linear array stereo images for mapping ap-

plications," *International Journal of Remote Sensing*, vol. 23, no. 20, pp. 4451–4473, 2002.

[26] D. Poli, "Modelling of spaceborne linear array sensors," Ph.D. dissertation, Eng., Politecnico di Milano, 2005.

[27] ——, "A rigorous model for spaceborne linear array sensors," *Photogrammetric Engineering and Remote Sensing*, vol. 73, no. 2, pp. 187–196, 2007.

[28] J. Shan, J. Yoon, D. Lee, R. Kirk, G. Neumann, and C. Acton, "Photogrammetric analysis of the Mars Global Surveyor mapping data," *Photogrammetric Engineering & Remote Sensing*, vol. 71, no. 1, 2005.

[29] M. Spiegel, U. Stilla, and G. Neukum, "Improving the exterior orientation of Mars Express regarding different imaging cases," in *Int. Arch. Photogramm. Remote Sensing Spatial Inf. Sci. XXXVI (Part 4A)*, 2006, p. 358363.

[30] R. Li, J. W. Hwangbo, Y. Chen, and K. Di, "Rigorous photogrammetric processing of HiRISE stereo images for mars topographic mapping," in *The XXI Congress of the International Society for Photogrammetry and Remote Sensing*, Beijing, China, 2008, pp. 987–992.

[31] J. Hwangbo, Y. Chen, and R. Li, "Precision processing of HiRISE stereo orbital images for topographic mapping on Mars," in *In proceedings of the ASPRS annual conference*, 2010.

[32] M. Bryson and S. Sukkarieh, "Observability analysis and active control for airborne SLAM," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 44, no. 1, pp. 261 –280, january 2008.

[33] V. Sazdovski and P. M. G. Silson, "Inertial navigation aided by vision based simultaneous localization and mapping," *Sensors Journal, IEEE*, no. 99, p. 1, 2011.

[34] J.-H. Kim and S. Sukkarieh, "Airborne simultaneous localisation and map building," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 2003, pp. 406–411.

[35] J. Artieda, J. M. Sebastian, P. Campoy, J. F. Correa, I. F. Mondragón, C. Martínez, and M. Olivares, "Visual 3-D SLAM from UAVs," *J. Intell. Robotics Syst.*, vol. 55, pp. 299–321, August 2009.

[36] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Vision-based odometry and SLAM for medium and high altitude flying UAVs," *Journal of Intelligent and Robotic Systems*, vol. 54, pp. 137–161, 2009.

[37] M. Bloesch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. of The IEEE International Conference on Robotics and Automation (ICRA)*, May 2010.

[38] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart, "Vision based position control for mavs using one single circular landmark," *Journal of Intelligent and Robotic Systems*, vol. 61, p. 495512, 2011.

[39] G. Nuetzi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and vision for absolute scale estimation in monocular SLAM," *Journal of Intelligent and Robotic Systems*, vol. 61, p. 287299, 2011.

[40] N. R. A. Bachrach, R. He, "Autonomous flight in unknown indoor environments," *International Journal of Micro Air Vehicles*, vol. 1, no. 4, pp. 217–228, December 2009.

[41] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *In Proceedings of the International Symposium on Experimental Robotics*, Delhi, India, Dec 2010, To Appear.

[42] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[43] J. M. M. M. Hauke Strasdat and A. J. Davison, "Real-time monocular SLAM: Why filter?" in *In Proceedings of International Conference on Artificial Reality and Telexistence*, Adelaide, SA, Australia, December 2010.

[44] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *CVPR*, 2010, pp. 1498–1505.

[45] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Proceedings of the 32nd DAGM conference on Pattern recognition*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 11–20.

[46] N. Birkbeck, D. Cobzas, and M. Jagersand, "Depth and scene flow from a single moving camera," in *3DPVT10*, 2010.

[47] P. Mountney, D. Stoyanov, A. J. Davison, and G.-Z. Yang, "Simultaneous stereoscope localization and soft-tissue mapping for minimal invasive surgery," in *MICCAI (1)*, 2006, pp. 347–354.

[48] M. Jama, C. Lewis, and D. Schinstock, "Identifying degrees of freedom in pushbroom bundle adjustment," *ISPRS Journal of Photogrammetry and Remote Sensing*, in press.

[49] ——, "Stereo processing pushbroom images with correlated path measurements," in *In proceedings of the ASPRS annual conference*, 2009.

[50] ——, "Analyzing scene geometries for stereo pushbroom imagery," in *In proceedings of the ASPRS annual conference*, 2010.

[51] OpenPilot, "OpenPilot Wikipedia," Internet, 2011, http://wiki.openpilot.org.

[52] M. Jama and D. Schinstock, "Parallel tracking and mapping for controlling VTOL airframe," *Journal of Intelligent & Robotic Systems*, under review.

[53] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Department of Computer Science, Stanford University, 1980.

[54] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[55] J. Shi and C. Tomasi, "Good features to track," in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, 1994, pp. 593 – 600.

[56] T. Lindeberg, *Scale-Space Theory in Computer Vision*. Norwell, MA, USA: Kluwer Academic Publishers, 1994.

[57] ——, "Scale-space theory: A basic tool for analysing structures at different scales," *Journal of Applied Statistics*, pp. 224–270, 1994.

[58] ——, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, pp. 79–116, 1998.

[59] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[60] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[61] S. M. Smith and J. M. Brady, "Susan a new approach to low level image processing," *Int. J. Comput. Vision*, vol. 23, pp. 45–78, May 1997.

[62] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, vol. 1, May 2006, pp. 430–443.

[63] R. Fletcher, *Practical Methods of Optimization*. Wiley, 2000.

[64] H. V. Poor, *An Introduction to Signal Detection and Estimation*. New York, NY: Springer-Verlag, 1994.

[65] J. Weng, N. Ahuja, and T. Huang, "Closed-form solution+maximum likelihood: a robust approach to motion and structure estimation," in *Computer Society Conference on Computer Vision and Pattern Recognition*, 1988, pp. 381–386.

[66] A. Blake and A. Zisserman, *Visual Reconstruction*. Cambridge, MA, USA: MIT Press, 1987.

[67] O. Hofmann, H. Ebner, and P. Nave, "DPS - a digital photogrammetric system for producing digital elevation models and orthophotos by means of linear-array scanner imagery," *Photogrammetric engineering and remote sensing*, vol. 50, no. 8, pp. 1135–1142, 1984.

[68] Z. Moore, D. Wright, D. E. Schinstock, and C. Lewis, "Comparison of bundle adjustment formulations," in *In proceedings of the ASPRS Annual Conference*, Baltimore, USA, 2009.

[69] B. Henderson, "A novel method for extrinsic self calibration of wide-baseline three-dimensional computer vision systems," Master's thesis, The University of New Mexico, 2006.

[70] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

[71] H. Ebner and G. Strunz, "Combined point determination using digital terrain models as control information," *International Archives of Photogrammetry and Remote Sensing*, vol. 27, no. 3, pp. 578–587, 1988.

[72] A. Hirano, R. Welch, and H. Lang, "Mapping from ASTER stereo image data: DEM validation and accuracy assessment," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 57, no. 5, pp. 356–370, 2003.

[73] ASTER GDEM Validation Team, "ASTER global DEM validation," METI, NASA, USGS, Tech. Rep., 2009, http://asterweb.jpl.nasa.gov/gdem.asp.

[74] J. Korona, E. Berthier, M. Bernard, F. Rmy, and E. Thouvenot, "SPIRIT. SPOT 5 stereoscopic survey of polar ice: Reference images and topographies during the fourth international polar year (2007-2009)," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 2, pp. 204 – 212, 2009.

[75] J.-P. Gleyzes, A. Meygret, C. Fratter, C. Panem, S. Baillarin, and C. Valorge, "SPOT5: system overview and image ground segment," in *Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International*, vol. 1, 2003, pp. 300–302.

[76] G. Chin, S. Brylow, M. Foote, J. Garvin, J. Kasper, J. Keller, M. Litvak, I. Mitrofanov, D. Paige, K. Raney, M. Robinson, A. Sanin, D. Smith, H. Spence, P. Spudis, S. A. Stern, and M. Zuber, "Lunar Reconnaissance Orbiter overview: The instrument suite and mission," *Space Science Reviews*, vol. 129, no. 4, pp. 391–419, 2007.

[77] J. Haruyama, T. Matsunaga, M. Ohtake, T. Morota, C. Honda, Y. Yokota, M. Torii, Y. Ogawa, and LISM Working Group, "Global lunar-surface mapping experiment using the lunar imager/spectrometer on SELENE," *Earth Planets Space*, vol. 60, no. 4, pp. 243–255, 2008.

[78] J. Haruyama, M. Ohtake, T. Matunaga, T. Morota, C. Honda, Y. Yokota, Y. Ogawa, and LISM Working Group, "SELENE (KAGUYA) terrain camera observation results of nominal mission period," in *40th Lunar and Planetary Science Conference*, 2009.

[79] A. Bouillon, M. Bernard, P. Gigord, A. Orsoni, V. Rudowski, and A. Baudoin, "Spot 5 hrs geometric performances: Using block adjustment as a key issue to improve quality of DEM generation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 60, no. 3, pp. 134 – 146, 2006.

[80] H. Ebner, W. Kornus, T. Ohlhof, and E. Putz, "Orientation of MOMS-02/D2 and MOMS-2P/PRIRODA imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54, no. 5-6, pp. 332 – 341, 1999.

[81] F. Ackermann, J. Bodechtel, F. Lanzl, D. Meissner, P. Seige, H. Winkenbach, and J. Zilger, "MOMS-02 a multispectral stereo scanner for the second german spacelab mission D2," in *International Geoscience and Remote Sensing Symposium, 1991. IGARSS '91.*, vol. 3, 1991, pp. 1727–1730.

[82] R. Sandau, H. D. Bernhard Braunecker, A. Eckardt, S. Hilbert, J. Hutton, W. Kirchhofer, E. Lithopoulos, R. Reulke, and S. Wicki, "Design principles of the LH systems ADS40 airborne digital sensor," *International Archives of Photogrammetry and Remote Sensing*, vol. 33, part B2, pp. 258–265, 2000.

[83] H.-P. Röser, A. Eckardt, M. von Schnermark, R. Sandau, and P. Fricker, "New potential and applications of ADS," *International Archives of Photogrammetry and Remote Sensing*, vol. 33, part B1, pp. 251–257, 2000.

[84] A. Delemere, I. Becker, J. Bergstrom, J. Burkepile, J. Day, D. Dorn, D. Gallagher, C. Hamp, J. Lasco, B. Meiers, A. Sievers, S. Streetman, S. Tarr, M. Tommeraasen, and P. Volmer, "MRO High Resolution Imaging Science Experiment (HIRISE): Instrument Development," in *Sixth International Conference on Mars*, 2003.

[85] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[86] B. Semenov, "HIRISE instrument kernel, ver. 1.1," February 2009, ftp://naif.jpl.nasa.gov/pub/naif/MRO/kernels/ik/mro_hirise_v11.ti (kernel fetched May, 25 2009).

[87] R. Salvini, M. Anselmi, A. Rindinella, and I. Callegari, "Quickbird stereo - photogrammetry for geological mapping (Cyrene-Libya)," in *ISPRS Congress Istanbul*, 2004.

[88] DigitalGlobe, "QuickBird spacecraft data sheet manual," Internet, 2009, http://www.digitalglobe.com/index.php/85/QuickBird.

[89] P. Fricker, "ADS40 progress in digital aerial data collection," in *In: D. Fritsch, R. Spiller (Eds.), Photogrammetric Week '01.* Heidelberg: Wichmann Verlag, 2001.

[90] R. Sandau, *Digital Airborne Camera.* Springer Netherlands, 2010.

[91] G. Neukum, "The airborne HRSC-A: Performance results and application potential," in *In D. Fritsch and R. Spiller (eds) Photogrammetric Week 99.* Wichmann Verlag, 1999.

[92] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Proc. 10th European Conference on Computer Vision (ECCV'08)*, Marseille, October 2008, pp. 802–815.

[93] R. O. Castle, G. Klein, and D. W. Murray, "Video-rate localization in multiple maps for wearable augmented reality," in *Proc 12th IEEE Int Symp on Wearable Computers, Pittsburgh PA, Sept 28 - Oct 1, 2008*, 2008, pp. 15–22.

[94] O. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 3, pp. 485–508, 1988.

[95] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 943–948.

[96] ——, "Homography-based 2D visual tracking and servoing," *Special Joint Issue on Robotics and Vision. Journal of Robotics Research*, vol. 26, no. 7, pp. 661–676, July 2007.

[97] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, October 2009, pp. 83 –86.

[98] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

Appendix A

Rotations

A.1 Rotation representations and transformations between them

Rotation matrix is a matrix that is used in an Euclidean coordinate space to perform rotation. Rotation matrix can represent rotation between two coordinate frames and can be used to transform point coordinates expressed in one frame to this point coordinates expressed in the other frame. For three dimensional space rotation matrix $\boldsymbol{R}$ is 3 by 3 matrix

$$\boldsymbol{R} = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} \\ R_{2,1} & R_{2,2} & R_{2,3} \\ R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix} \tag{A.1}$$

$\boldsymbol{R}$ has interesting properties, including its determinant being 1, all the rows and columns being orthonormal, and the inverse being equal to identity, i.e., $R^{-1} = R^T$.

Rotation matrix is one way of representing this rotations, others include Euler angles and quaternions. This section provides formulas to transform between different representation of the same rotation.

A.1.1 Euler angles

Each rotation can be expressed as a composition of rotations along X, Y, and Z axes. Assuming that we rotate by α, β, and γ along X, Y, and Z respectively we have

$$\boldsymbol{R}_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & s_\alpha \\ 0 & -s_\alpha & c_\alpha \end{bmatrix} \qquad \boldsymbol{R}_Y = \begin{bmatrix} c_\beta & 0 & -s_\beta \\ 0 & 1 & 0 \\ s_\beta & 0 & c_\beta \end{bmatrix} \qquad \boldsymbol{R}_Z = \begin{bmatrix} c_\gamma & s_\gamma & 0 \\ -s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $c_* = \cos(*)$ and $s_* = \sin(*)$.

Those three rotations can be combined in any order, but two arrangements are the most common. If we rotate along Z axis first, then along new Y axis, and finally along new X axis the corresponding rotation matrix $\mathbf{R}$ is given as:

$$\begin{aligned} \mathbf{R} &= \boldsymbol{R}_X \boldsymbol{R}_Y \boldsymbol{R}_Z \\ &= \begin{bmatrix} c_\beta c_\gamma & c_\beta s_\gamma & -s_\beta \\ -c_\alpha s_\gamma + s_\alpha s_\beta c_\gamma & c_\alpha c_\gamma + s_\alpha s_\beta s_\gamma & s_\alpha c_\beta \\ s_\alpha s_\gamma + c_\alpha s_\beta c_\gamma & -s_\alpha c_\gamma + c_\alpha s_\beta s_\gamma & c_\alpha c_\beta \end{bmatrix} \end{aligned}$$

This arrangement is commonly used in aviation and three angles, α, β and *omega*, are called *roll*, *pitch* and *yaw* respectively. Given rotation matrix $\mathbf{R}$ we can find associated *roll*, *pitch* and *yaw*

$$\alpha = \arctan(R_{2,3}, R_{3,3}) \qquad \beta = \arcsin(-R_{1,3}) \qquad \gamma = \arctan(R_{1,2}, R_{1,1})$$

On the other hand if we rotate along X axis first, then along new Y axis and finally along new Z axis we receive following rotation matrix

$$\begin{aligned} \mathbf{R} &= \boldsymbol{R}_Z \boldsymbol{R}_Y \boldsymbol{R}_X \\ &= \begin{bmatrix} c_\phi c_\kappa & c_\omega s_\kappa + s_\omega s_\phi c_\kappa & s_\omega s_\kappa - c_\omega s_\phi c_\kappa \\ -c_\phi s_\kappa & c_\omega c_\kappa - s_\omega s_\phi s_\kappa & s_\omega c_\kappa + c_\omega s_\phi s_\kappa \\ s_\phi & -s_\omega c_\phi & c_\omega c_\phi \end{bmatrix} \end{aligned}$$

This convention is prominent in photogrammetry field where having the last rotation along the Z axis (usually camera boresight) simplifies some computations. In this context angles are usually marked as ω, ϕ and κ. Again, given rotation matrix $\mathbf{R}$ we can recover ω, ϕ and κ

$$\omega = \arctan(-R_{3,2}, R_{3,3}) \qquad \phi = \arcsin(R_{3,1}) \qquad \kappa = \arctan(-R_{2,1}, R_{1,1})$$

A.1.2 Quaternions

Any complex rotation can be expressed as a single rotation around a vector. We define θ to be magnitude of this single rotation around an unit vector $\boldsymbol{r} = [r_1, r_2, r_3]^T$. Quaternion is then expressed as:

$$\begin{aligned}\mathbf{q} &= [q_0, q_1, q_2, q_3]^T \\ &= \left[\cos\left(\frac{\theta}{2}\right), r_1 \sin\left(\frac{\theta}{2}\right), r_2 \sin\left(\frac{\theta}{2}\right), r_3 \sin\left(\frac{\theta}{2}\right)\right]^T\end{aligned}$$

Any rotation matrix $\boldsymbol{R}$ can be written as a function of a quaternion $\boldsymbol{q}$ as $\boldsymbol{R}(\boldsymbol{q})$ where $R_{i,j}$ elements of Eq. A.1 are given as

$$\begin{aligned} R_{1,1} &= q_0^2 + q_1^2 - q_2^2 - q_3^2 & R_{1,2} &= 2(q_1 q_2 + q_0 q_3) & R_{1,3} &= 2(q_1 q_3 - q_0 q_2) \\ R_{2,1} &= 2(q_1 q_2 - q_0 q_3) & R_{2,2} &= q_0^2 - q_1^2 + q_2^2 - q_3^2 & R_{2,3} &= 2(q_2 q_3 + q_0 q_1) \\ R_{3,1} &= 2(q_1 q_3 + q_0 q_2) & R_{3,2} &= 2(q_2 q_3 - q_0 q_1) & R_{3,3} &= q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{aligned}$$

A.2 Finding rotation between two frames

The problem is formulated as follows: given set of n corresponding rotations $R^i_{c,p}$ and $R^i_{eb,ef}$, where $i \in [1 \dots n]$, find constant rotations $R_{ef,p}$ and $R_{c,eb}$. Note that it is enough to find either $R_{ef,p}$ or $R_{c,eb}$ and the other will follow trivially. Here, we first solve for $R_{c,eb}$. All the rotations are presented in Fig. A.1. We start by formulating a vector loop:

$$R_{ef,p} = R_{ef,eb} R_{eb,c} R_{c,p}$$

Now, since $R_{ef,p}$ is constant, i.e., $R^i_{ef,p} = R^j_{ef,p}$ for any $i, j \in [1 \dots n]$ we can write

$$R^i_{ef,eb} R_{eb,c} R^i_{c,p} = R^{i+1}_{ef,eb} R_{eb,c} R^{i+1}_{c,p}$$

Above equation can be used to formulate elements of the cost function $\boldsymbol{c}(\boldsymbol{K})$ where $\boldsymbol{K}$ is a parameter vector. $\boldsymbol{K}$ can be chosen to be three Euler angles representing the rotation $R_{eb,c}$

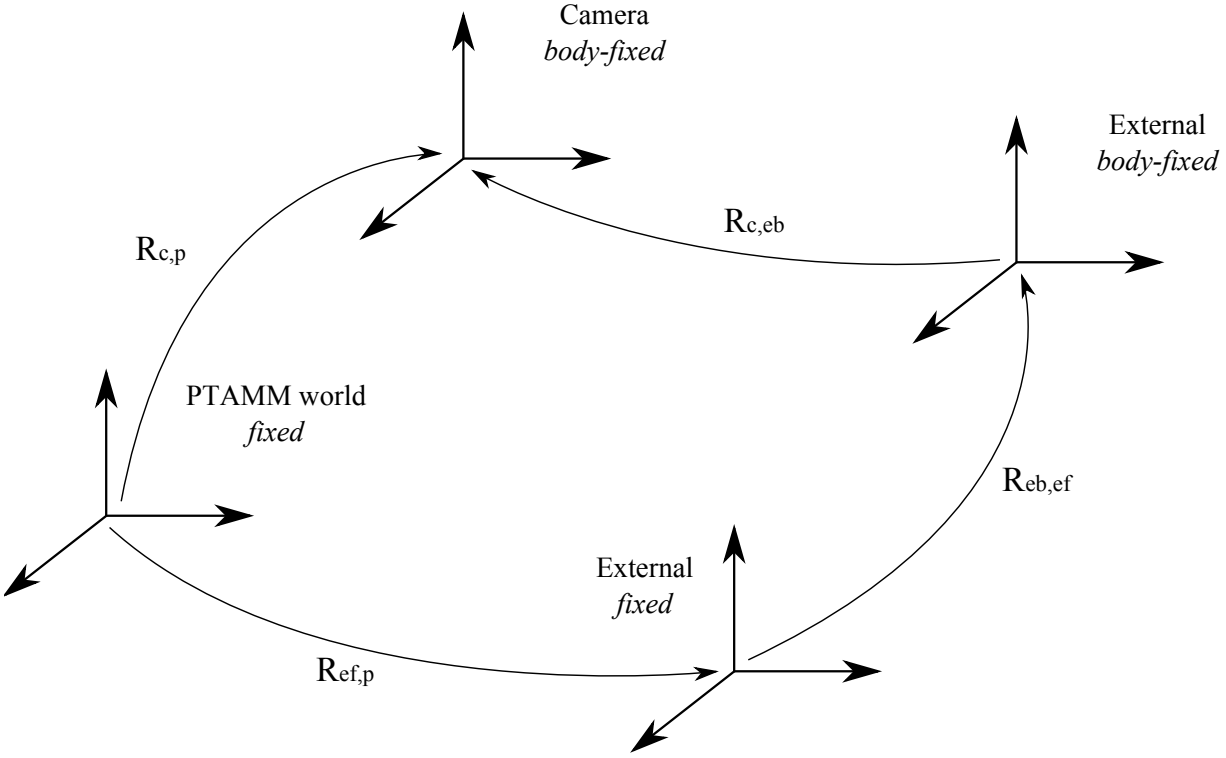


Figure A.1: *Coordinate frames*

and then the cost function becomes

$$\boldsymbol{c} = \begin{bmatrix} R^0_{ef,eb} R_{eb,c} R^0_{c,p} - R^1_{ef,eb} R_{eb,c} R^1_{c,p} \\ R^1_{ef,eb} R_{eb,c} R^1_{c,p} - R^2_{ef,eb} R_{eb,c} R^2_{c,p} \\ \cdots \\ R^{n-1}_{ef,eb} R_{eb,c} R^{n-1}_{c,p} - R^n_{ef,eb} R_{eb,c} R^n_{c,p} \\ R^n_{ef,eb} R_{eb,c} R^n_{c,p} - R^0_{ef,eb} R_{eb,c} R^0_{c,p} \end{bmatrix}$$

However, singularities in this representation may cause problem in the minimization. We choose to use $R_{eb,c}$ as $\boldsymbol{K}$ directly. To ensure minimization is converging to a matrix that is a rotation matrix additional elements enforcing matrix rows to be orthogonal unit vectors need to be added to the cost function. Using defined cost function the problem can be solved by using minimization algorithm e.g. Levenberg-Marquard.