ABSTRACT

Title of dissertation: VIDEO PROCESSING WITH ADDITIONAL INFORMATION

Mahesh Ramachandran, Doctor of Philosophy, 2010

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

Cameras are frequently deployed along with many additional sensors in aerial and ground-based platforms. Many video datasets have metadata containing measurements from inertial sensors, GPS units, etc. Hence the development of better video processing algorithms using additional information attains special significance.

We first describe an intensity-based algorithm for stabilizing low resolution and low quality aerial videos. The primary contribution is the idea of minimizing the discrepancy in the intensity of selected pixels between two images. This is an application of inverse compositional alignment for registering images of low resolution and low quality, for which minimizing the intensity difference over salient pixels with high gradients results in faster and better convergence than when using all the pixels.

Secondly, we describe a feature-based method for stabilization of aerial videos and segmentation of small moving objects. We use the coherency of background motion to jointly track features through the sequence. This enables accurate tracking of large numbers of features in the presence of repetitive texture, lack of well conditioned feature windows etc. We incorporate the segmentation problem within the joint feature tracking

framework and propose the first combined joint-tracking and segmentation algorithm. The proposed approach enables highly accurate tracking, and segmentation of feature tracks that is used in a MAP-MRF framework for obtaining dense pixelwise labeling of the scene. We demonstrate competitive moving object detection in challenging video sequences of the VIVID dataset containing moving vehicles and humans that are small enough to cause background subtraction approaches to fail.

Structure from Motion (SfM) has matured to a stage, where the emphasis is on developing fast, scalable and robust algorithms for large reconstruction problems. The availability of additional sensors such as inertial units and GPS along with video cameras motivate the development of SfM algorithms that leverage these additional measurements. In the third part, we study the benefits of the availability of a specific form of additional information - the vertical direction (gravity) and the height of the camera both of which can be conveniently measured using inertial sensors, and a monocular video sequence for 3D urban modeling. We show that in the presence of this information, the SfM equations can be rewritten in a bilinear form. This allows us to derive a fast, robust, and scalable SfM algorithm for large scale applications. The proposed SfM algorithm is experimentally demonstrated to have favorable properties compared to the sparse bundle adjustment algorithm. We provide experimental evidence indicating that the proposed algorithm converges in many cases to solutions with lower error than state-of-art implementations of bundle adjustment. We also demonstrate that for the case of large reconstruction problems, the proposed algorithm takes lesser time to reach its solution compared to bundle adjustment. We also present SfM results using our algorithm on the Google StreetView research dataset, and several other datasets.

VIDEO PROCESSING WITH ADDITIONAL INFORMATION

by

Mahesh Ramachandran

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2010

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor Ankur Srivastava
Professor Richard La
Professor David Jacobs
Professor Min Wu

I dedicate this dissertation to my parents

and my grandfather, Mr. P. P. Iyer.

Acknowledgments

I am deeply indebted to my advisor, Prof. Rama Chellappa, for being a constant source of support, guidance and funding through my doctoral studies. Through these years, I am lucky to have worked with Prof. Chellappa whom I have always deeply admired. He believed in me and my potential through so many tough times that helped me gain confidence to wade through rough waters and finally reach the destination. I had more than a fair share of problems towards the start of my Ph.D, and his continued support helped me get over them. Around the beginning of my doctoral studies, I had concentration and memory problems linked to a deficit of attention, and I had to seek professional and medical help. This particularly rough time, soon after I had stepped into a new country made me wonder if I had made the right choice. My decision to fight on and a never-say-die attitude was one of the best decisions I have made, and the culmination of my efforts, resulting in my PhD, would not have happened without Prof. Chellappa's encouragement and dissertation advise.

I would also like to thank medical doctors Dr. Krishnamoorthy Srinivas (India), Dr. Joseph Marnell, Dr. Punja Sudhakar, and Dr. David Granite for helping me get through the problems with my concentration. I was much better off with their advise and treatment.

I specially thank Ashok for his collaboration and support through my doctoral studies. He helped me as a friend and as a collaborator, and I am really grateful for everything I have learnt from him. My close interactions with him helped me look at problems from various points of view and sharpen my problem solving approach. I admire his clear

thought process and his novel approach to research problems.

I thank my committee members, Prof. David Jacobs, Prof. Ankur Srivastava, Prof. Min Wu and Prof. Richard La for their comments and questions about my dissertation work, which helped me look at my work from various viewpoints.

I am fortunate to have spent my graduate school years with three exceptional friends, Ashwin, Krishna and Aswin, whom I have known from my time at IIT Madras. They were extremely supportive and helpful through my PhD studies. Special thanks to Ashwin (whom I have known from high school) for being my roommate all through graduate school. I have benefitted immensely through my brainstorming sessions with Ashwin and Aswin while at UMD.

I thank all the members in CFAR who helped me directly or indirectly. Aravind was particularly helpful with his sound advise. Thanks to Kaushik for being always available to bounce ideas. I also thank Jagan, Kevin, Jian Li, Jie Shao, Zhanfeng, Yang, James, Narayanan, Gaurav, Soma, Hao, Pavan, Dikpal, Mohammad, Seong-Wook, Sima, Jai, Ming, Ming-Yu, Vishal for helping me on various occasions.

My friends at Greenbelt made my stay enjoyable and gave me good company off work hours. Ashwin, Sebastian, Ranjit, Rajesh, Alankar were a great group to hang out with and I have shared many memorable times with them. I also enjoyed being housemates with Sidharth and Sowmitra at various times.

This dissertation would not have been possible without the constant and unending support of my family. I have been fortunate to have a family who always believed in me and encouraged me in my pursuit of excellence. My love and gratitude to my parents, Ramachandran (appa) and Girija (ammma), my kid brother Ganesh, my cousins Anusha,

Arvind and Ravi, Naganathan chittappa and Radha chitti, my grandfathers P.P.Iyer (appappa) and Ganapati (sugar thatha) and grandmother Thailambal thathi. I also thank Dr. Ramani Ramchandran and Ashley Skinner for being there whenever I needed them, during the initial years of my graduate life. I have always enjoyed the thanksgiving parties they organized at their home, to which I was invited by default.

I dedicate this dissertation to my parents and my grandfather, Mr. P. P. Iyer. Appappa (which in tamil means father's father) was my role model through my childhood years and was instrumental in my development and education. He was a great man and a true visionary.

Table of Contents

List of Figures

List of Abbreviations

3D	3 Dimensions
BA	Bundle Adjustment
CCS	Camera Coordinate System
CG	Conjugate Gradient
FBSfM	Fast Bilinear Structure from Motion
GCT	Global Convergence Theorem
GPN	Ground Plane Normal
GPS	Global Positioning System
IMU	Inertial Measurement Unit
KLT	Kanade Lucas Tomasi
LM	Levenberg-Marquardt
MAP	Maximum A-Posteriori
MAV	Micro Air Vehicle
MRF	Markov Random Field
RANSAC	Random Sampling And Consensus
SBA	Sparse Bundle Adjustment
SfM	Structure from Motion
SfPM	Structure from Planar Motion
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SVD	Singular Value Decomposition
UAV	Unmanned Air Vehicle
UTM	Universal Transverse Mercator
VIVID	VIdeo Verification and IDentification
WCS	World Coordinate System

Chapter 1

Introduction

With advances in the development of Unmanned and Micro air vehicles, there is an increasing need to process the video streams obtained from cameras on-board these aerial platforms. Due to the erratic motion of these aerial platforms, robust stabilization of these video sequences is an important video pre-processing task. In addition, acquisition of low-resolution video sequences makes it necessary to mosaic the video sequences for better visualization using a larger virtual field-of-view [3]. Generating mosaics of the scene involves the detection and removal of moving object pixels, and hence robust moving object detection becomes important.

Video stabilization refers to the compensation of the motion of pixels on the image plane, when a video sequence is captured from a moving camera. Stabilization of a video sequence is achieved by registering consecutive images of the sequence onto each other. A parametric motion model such as an affine or homography model is assumed for the image sequence and the model parameters are estimated for every pair of consecutive frames in the image sequence. The pairwise models are used to compute the transformations registering each frame onto a global reference or mosaic.

Image registration is a widely researched topic in computer vision, with applications in mosaicing, medical imaging, motion detection, tracking etc. The two main approaches for image registration are feature-based and intensity-based approaches. In the feature-

based approach, we identify corresponding image features (such as points and lines) in an image sequence. Using these feature correspondences, we solve for the motion parameters that register the images. In intensity-based approaches, we solve for the motion parameters by minimizing the intensity difference between the two images.

A video sequence acquired by an aerial platform consists of multiple moving targets of interest moving in and out of the field-of-view of the camera. An important problem involves the segmentation of the moving object pixels from the background. This is useful both for mosaicing the background as well as moving target initialization for tracking. An understanding of the motion of targets in video is necessary for several video inference tasks such as activity analysis, video summarization etc.

Moving target pixels do not obey the solved background motion model after video stabilization and do not register in consecutive frames. Moving targets are typically detected by identifying the subset of pixels that are not aligned after video stabilization, therefore motion detection results depend on the accuracy of video stabilization. Highly accurate stabilization is extremely important for detecting small objects moving in similar background texture.

Structure from Motion (SfM) refers to the task of recovering the 3D structure of a scene and the motion of a camera from a video sequence. SfM has been an active area of research since Longuet-Higgins [4] eight-point algorithm. There have been several different approaches to the SfM problem and we refer the reader to [5] and [6] for a comprehensive survey of the various approaches.

1.1 Video Stabilization

Video stabilization refers to the compensation of the motion of pixels on the image plane, when a video sequence is captured from a moving camera. Since several practical applications just require video stabilization as opposed to complete estimation of camera trajectory and scene structure, this is an important sub-problem that has received much attention. Depending on the type of scenario and the type of motion involved, we have different algorithms to achieve stabilization.

- *Presence of a dominant plane in the scene:* If a dominant plane is present, then we can register all the frames using a planar perspective transformation (homography) corresponding to that plane. For pixels that do not lie on the plane, we need to warp them appropriately depending on the amount of parallax. This is a very common assumption for aerial videos and surveillance cameras monitoring a scene with a dominant ground-plane.

- *Derotation of the image sequence:* In some applications [7, 8], we may want to estimate and remove the motion due to only the 3D rotation of the camera. This corresponds to derotation of the image sequence.

- *Mosaic construction:* We may need to build an extended field-of-view mosaic of the scene using images in the sequence. In this case, we need to accurately register and blend the various images onto the mosaicing surface.

- *Presence of moving objects:* One objective of stabilization is to register the video frames and segment out the moving objects from the scene. This involves detecting

independent motion that is different from ego-motion.

In this dissertation, we address the problems of stabilization and motion detection in aerial videos. Our datasets consist of sequences obtained from low-quality and low-resolution cameras which have very few prominent features in them, or sequences where feature tracking is inherently a difficult problem because of repeated textures etc. In addition, our objective is to detect moving vehicles and people in aerial videos when these objects occupy a small area in the image.

1.2 Structure from Motion

The problem of SfM has gained renewed interest because of exciting new applications like 3D urban modeling, terrain estimation from UAVs and photo-tourism. Companies like Google and Microsoft, through their StreetView and WindowsLive products, already support some applications such as large-scale urban visualization. Google has been collecting image sequences from omnidirectional cameras over thousands of miles of roads in several cities around the world. Through its StreetView application, it is now possible to look at these image sequences geo-located on a map. Figure 1.1(a) shows an image (downloaded from [1]) of a car with cameras attached to it, capturing images of a city while driving. Fast, robust and scalable SfM algorithms would enable automatic creation of 3D models of urban scenes from the sequence of images recorded using such mobile cameras. This would enrich the content in current digital maps and potentially provide 3D content with both structural information and textures overlaid.

To solve such large scale SfM applications, several technical challenges need to be

addressed. Firstly, most of these large scale image data collections usually involve the presence of additional sensors such as inertial measurement units, global positioning systems etc. Traditional SfM approaches need to be adapted to efficiently incorporate such additional sensor measurements into a consistent global estimation framework. Secondly, the scale of models that need to be built are much larger than ever before. This means that the algorithms developed for SfM must be fast, scalable and eminently parallelizable. We consider the problem of SfM estimation in the presence of a specific form of additional information that is frequently available, and propose a fast, scalable and robust SfM algorithm.

1.3 Contributions

This dissertation consists of several contributions to important video processing tasks such as stabilization, mosaicking, motion detection and structure from motion in aerial video sequences, that has resulted in a pipelined video processing system. Figure 1.2 illustrates the schematic of the pipelined video processing system. This dissertation makes algorithmic contributions in all the individual blocks shown in the figure 1.2. The list of contributions is as follows.

- We study the intensity-based alignment algorithm and its application for the stabilization of low quality image sequences. We demonstrate that using only a subset of salient pixels for registration, we can get better registration accuracies in lesser computation time. This study is new, and has implications for registration algorithms deployed on small devices with limited computational power.

(a) A car with attached cameras

(b) A helicopter with cameras and inertial sensors

(c) A UAV surveying the ground plane

Figure 1.1: The figure shows illustrations of the problem domains where our work is applicable. Figure (1.1(a)) shows an image of a car with a set of cameras attached to the mount on top, collecting image sequences of the urban scene. Figure (1.1(b)) shows a helicopter sensing the environment with cameras, IMUs and other sensors. Figure (1.1(c)) illustrates a UAV surveying the ground plane from a high altitude. These images were downloaded from [1]. In all these cases, we have available additional information along with the sequences that can be used in our algorithmic framework.

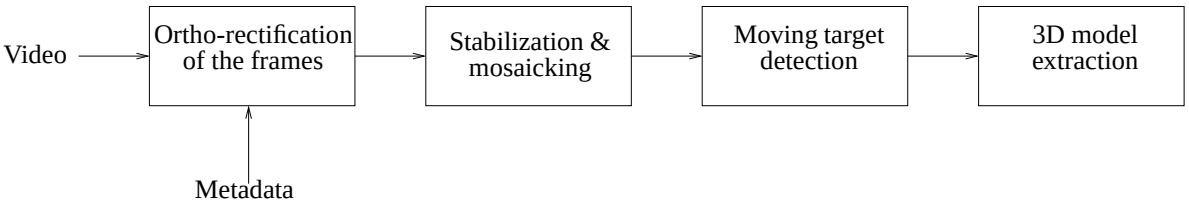


Figure 1.2: An illustration of the pipelined video system developed as part of the work presented in this dissertation.

- We propose a joint tracking and segmentation algorithm to exploit motion coherency of the background as well as solve for the class labels of features. Although algorithms exist for tracking features jointly, the idea of incorporating segmentation within this framework and using the feature dissimilarities to infer membership

probabilities is new. The proposed approach produces highly accurate labeled feature tracks in a sequence that are uniformly distributed. This enables us to infer dense pixelwise motion segmentation that is useful in moving target initialization.

- We demonstrate competent stabilization and motion detection results in several challenging video sequences in the VIVID dataset. We also qualitatively compare the improvement in image mosaics obtained using the information provided by associated metadata.

- We consider the problem of SfM estimation in the presence of a specific form of additional information that is frequently available, and propose a fast, scalable and robust SfM algorithm that is bilinear in the Euclidean frame.

- We describe simulation results demonstrating that the proposed algorithm leads to solutions with lower error than SBA and takes lower time for convergence.

- We describe competitive reconstruction results on the Google StreetView research dataset.

1.4 Outline

This dissertation is organized as follows.

In chapter 2, we provide the background theory in computer vision that is fundamental to the rest of the dissertation. We describe important camera and motion models and describe how camera motion induces motion in the image. We describe how image features extracted from image sequences are related among multiple views, and describe

the general methodology of feature-based algorithms. We review related work in stabilization, mosaicking and structure from motion.

In chapter 3, we describe our work on intensity-based stabilization and present our study on how a fraction of image pixels can be used for better registration at lower computational cost. We describe registration and mosaicking results on low quality and low resolution aerial datasets.

In chapter 4, we present our work on feature-based stabilization and motion segmentation. We describe our algorithm for joint feature tracking and segmentation and illustrate how it can be used for stabilizing video and detecting moving objects with high accuracy and very few false alarms. We present competitive results on challenging sequences from the VIVID dataset and describe how the metadata is used in our algorithm.

Chapter 5 addresses the problem of structure from motion using aerial or ground-based sequences with associated metadata. We present a fast, robust and scalable SfM algorithm that uses additional measurements about gravity and height to express the SfM equations in a bilinear form in the Euclidean frame. We analyze the computational complexity and memory requirements of the algorithm. We present extensive simulation results that illustrate the favorable properties of the algorithm compared to bundle adjustment in terms of speed. We present results on several real datasets such as VIVID, Google StreetView research dataset etc.

Chapter 6 presents several future directions of research and concludes the dissertation. Appendix A provides a proof of convergence of the proposed structure from motion algorithm and appendix B describes how to decompose a set of homographies induced by a plane in multiple views to obtain the plane normal vectors and heights that are useful

for our algorithm.

Chapter 2

Background and Prior Art

2.1 Background

Prior to discussing models for the global motion problem, it is worthwhile to verify whether the apparent motion on the image induced by the camera motion can indeed be approximated by a global model. This study takes into consideration an analytic model for the camera as a projective device, the 3D structure of the scene being viewed, and its corresponding image. We describe the model for a projective camera and study the how the image of a world point moves as the camera undergoes general motion (three translations and three rotations).

2.1.1 Camera Model

The imaging geometry of a perspective camera is shown in Fig. 2.1. The origin of the 3D coordinate system (X, Y, Z) lies at the optical center C of the camera. The *retinal plane* or *image plane* is normal to the optical axis Z, and is offset from C by the focal length f. Images of unoccluded 3D objects in front of the camera are formed on the image plane. The 2D image plane coordinate system (x, y) is centered at the *principal point*, which is the intersection of the optical axis with the image plane. The orientation of (x, y) is flipped with respect to (X, Y) in Fig. 2.1, due to inversion caused by simple transmissive optics. For this system, the image plane coordinate (x_i, y_i) of the image of

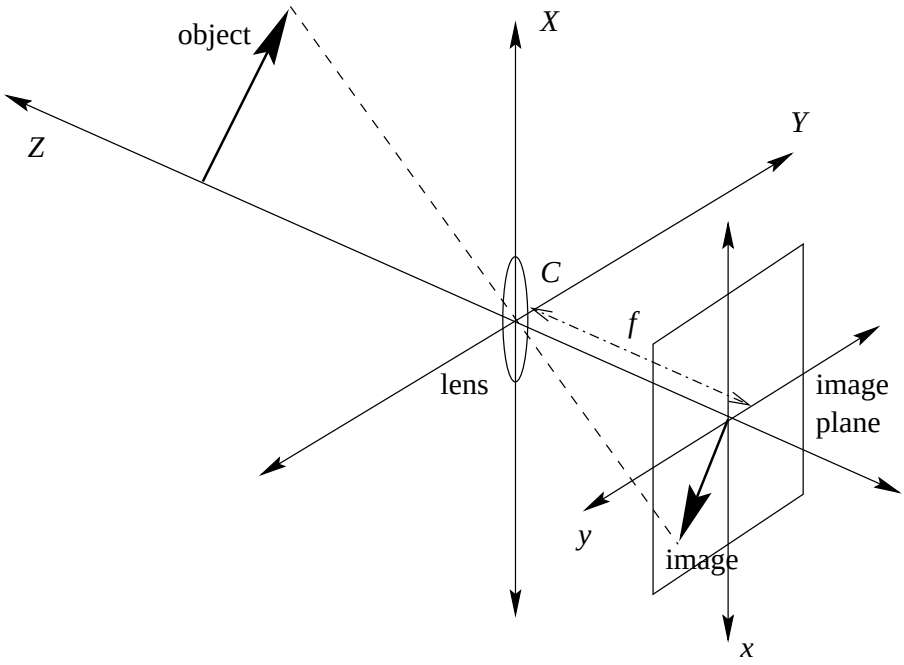


Figure 2.1: 3D imaging geometry.

the unoccluded 3D point (X_i, Y_i, Z_i) is given by

$$x_i = f\frac{X_i}{Z_i} \quad , \quad y_i = f\frac{Y_i}{Z_i} \; . \tag{2.1}$$

The projective relation (2.1) assumes a rectilinear system, with an isotropic optical element. In practice, the plane containing the sensor elements may be misaligned from the image plane, and the camera lens may suffer from optical distortions including non-isotropy. However, these effects can be compensated by calibrating the camera, and/or remapping the image. In the remainder of this chapter, it is assumed that the linear dimensions are normalized w.r.t. the focal length, i.e. $f = 1$.

2.1.2 Effect of Camera Motion

The effect of camera motion can be computed using projective geometry [9, 10]. Assume that an arbitrary point in the 3D scene lies at (X_0, Y_0, Z_0) in the reference frame of the first camera, and moves to (X_1, Y_1, Z_1) in the second. The effect of camera motion relates the two coordinate systems according to:

$$\begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} = \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{pmatrix} \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} , \tag{2.2}$$

where the rotation matrix $[r_{ij}]$ is a function of ω. Combining (2.1) and (2.2) permits the expression of the projection of the point in the second image in terms of that in the first as:

$$\begin{aligned} x_1 &= \frac{r_{xx}x_0 + r_{xy}y_0 + r_{xz} + t_x/Z_0}{r_{zx}x_0 + r_{zy}y_0 + r_{zz} + t_z/Z_0} , \\ y_1 &= \frac{r_{yx}x_0 + r_{yy}y_0 + r_{yz} + t_y/Z_0}{r_{zx}x_0 + r_{zy}y_0 + r_{zz} + t_z/Z_0} . \end{aligned} \tag{2.3}$$

Assuming either that (i) points are distant compared to the inter-frame translation, i.e. neglecting the effect of translation, or (ii) a planar embedding of the real world, the *perspective* transformation is obtained:

$$\begin{aligned} x_1 &= \frac{p_{xx}x_0 + p_{xy}y_0 + p_{xz}}{p_{zx}x_0 + p_{zy}y_0 + p_{zz}} , \\ y_1 &= \frac{p_{yx}x_0 + p_{yy}y_0 + p_{yz}}{p_{zx}x_0 + p_{zy}y_0 + p_{zz}} . \end{aligned} \tag{2.4}$$

Other popular global deformations mapping the projection of a point between two

frames are the similarity and affine transformations, which are given by:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = s \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} , \tag{2.5}$$

and

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} a_0 & a_1 \\ a_2 & a_3 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \tag{2.6}$$

respectively. Free parameters for the similarity model are the scale factor s, image plane rotation θ and translation (b_0, b_1). The affine transformation is a superset of the similarity operator, and incorporates shear and skew as well. The perspective operator is a superset of the affine, as can be readily verified by setting $p_{zx} = p_{zy} = 0$ in (2.4).

Next, we discuss how to extract features from images and how they can be used for computing the image motion using the models described earlier.

2.1.3 Image features

The basic goal in feature-based motion estimation is to use features to find maps that relate the images taken from different view-points. These maps are then used to estimate the image motion by computing the parameters of a motion model. Consider the case of pure rotation. Here, the camera center is fixed and the image plane is moved to another position. The image of a point in the real world is formed by the intersection on the image plane of the ray joining the camera center and the world point. The resulting images formed on the image planes are quite different but they are related in an interesting way.

Though various lengths, ratios, angles formed on the images are all different, the

cross ratio remains the same [11]. Given four collinear points A, B, C and D on an image, the cross ratio is $\frac{AC}{CB} \div \frac{AD}{DB}$ and it remains constant. In other words, $\frac{AC}{CB} \div \frac{AD}{DB} = \frac{\acute{A}\acute{C}}{\acute{C}\acute{B}} \div \frac{\acute{A}\acute{D}}{\acute{D}\acute{B}}$, where $\acute{A}$, $\acute{B}$, $\acute{C}$ and $\acute{D}$ are the corresponding points in the second image (formed after rotating the camera about its axis).

Looking carefully, we can see that this intuition leads to a map relating the two images. Given four corresponding points in general position in the two images, we can map any point from one image to the other. Suppose we know that A maps to $\acute{A}$, B to $\acute{B}$, C to $\acute{C}$ and D to $\acute{D}$. Then the point of intersection of AB and CD (say E) will map to the point of intersection of $\acute{A}\acute{B}$ and $\acute{C}\acute{D}$ (say $\acute{E}$). Now any point F on ABE will map to point $\acute{F}$ such that the cross ratio $\frac{AE}{EB} \div \frac{AF}{FB}$ is preserved. This way one can map each point from one image to the other image. Such a map is called a *homography*. As mentioned before, such a map is defined by four corresponding points in general position. So, if x maps to $\acute{x}$ by homography H, $x' = Hx$. Note that such a map exist only in case of pure rotation.

However, for planar scenes, homography relating the two views exist irrespective of the motion involved. In the case of planar scene, there exist a homography relating the first image to the real-world plane and another one mapping the real-world plane to the second image plane, i.e.,

$$x_1 = H_1 x_p \tag{2.7}$$

$$x_p = H_2 x_2 \tag{2.8}$$

$$\Rightarrow x_1 = H_1 H_2 x_2 = H x_2 \tag{2.9}$$

where H_1 maps x_1, a point on first image plane to x_p, the corresponding point on the

real plane while H_2 maps x_p to x_2, the corresponding point on the second image plane. Thus homography $H = H_1 H_2$ maps points from one image plane to the other. Such a homography exists, no matter what the underlying motion between the two camera positions is. This happens because the images formed by camera rotation (or in the case of planar scenes) do not depend on the scene structure. On the other hand, when there are depth variations in the scene, such a homography doesn't exist between images formed by camera translation.

In the case of depth variations, we can use SfM approaches to estimate the motion of the camera. The estimated camera motion can be used to selectively stabilize the image sequence for the camera motion (such as compensation of rotation, or sideways translation etc.) The next section discusses about SfM and describes an algorithm for SfM using image feature points.

2.1.4 Structure from Motion

Structure from motion refers to the task of inferring the camera motion and scene structure from an image sequence taken from a moving camera, using either image features or flow. We describe an illustrative algorithm for SfM using feature point tracks in a sequence. Consider an image sequence with N images. Feature points are detected and tracked throughout the sequence. Suppose the scene has M features and their projections in each image are denoted by $x_{i,j} = (u_{i,j}, v_{i,j})^T$ where $i \in \{1, \cdots, M\}$ denotes the feature index and $j \in \{1, \cdots, N\}$ denotes the frame index. For the sake of simplicity, assume that all features are visible in all frames. The structure-from-motion problem

involves solving for the camera locations and the 3D coordinates of feature points in the world coordinate system.

The camera poses are specified by a rotation matrix R_j and a translation vector T_j for $j = 1, \cdots, N$. The coordinates of a point in the camera system and world system are related by $P_c = R_i P_w + T_i$, where P_c denotes the coordinates of a point in the camera coordinate system, and P_w denotes the coordinates of the same point in the world coordinate system. The 3D coordinates of the world landmarks are denoted by $\underline{X}_i = (X_i, Y_i, Z_i)^T$ for $i = 1, \cdots, M$. We assume an orthographic projection model for the camera. Landmarks are projected onto the image plane according to the following equation:

$$\begin{pmatrix} u_{i,j} \\ \\ v_{i,j} \end{pmatrix} = K \cdot \begin{bmatrix} R_j & T_j \end{bmatrix} \underline{X}_i \tag{2.10}$$

In equation (2.10), K denotes the 2×3 camera matrix. Let the centroid of the $3D$ points be C and the centroid of the image projections of all features in each frame be c_j. We can eliminate the translations from these equations by subtracting out C from all world point locations and c_j from the image projections of all features in the j^{th} frame. Let $\hat{x}_{i,j} = x_{i,j} - c_j$ and $\hat{X}_j = \underline{X}_j - C$. The projection equation can be rewritten as:

$$\hat{x}_{i,j} = P_j \cdot \hat{X}_i \tag{2.11}$$

In equation (2.11), $P_j = K \cdot R_j$ denotes the 2×3 projection matrix of the camera.

We can stack up the image coordinates of the all the feature points in all the frames,

and write it in the factorization format as follows:

$$\begin{bmatrix} \hat{x}_{1,1} & \cdots & \hat{x}_{M,1} \\ \vdots & \ddots & \vdots \\ \hat{x}_{1,N} & \cdots & \hat{x}_{M,N} \end{bmatrix} = \begin{bmatrix} P_1 \\ \vdots \\ P_N \end{bmatrix} \begin{bmatrix} \hat{X}_1 & \cdots & \hat{X}_M \end{bmatrix} \tag{2.12}$$

The matrix on the left hand side of equation (2.12) is the measurement matrix, and it has been written as a product of a $2N \times 3$ projection matrix and a $3 \times M$ structure matrix. This factorization implies that the measurement matrix is of rank 3. This observation leads to a factorization algorithm for solving for the projection matrices P_j and the $3D$ point locations X_i. The details of the algorithm are described in detail in [12].

2.1.5 Feature based algorithms

We have seen that a homography can be used to map one image to the other in the case of pure camera rotation, or a planar scene. If such a homography exists between the images, four points are sufficient to specify it precisely. In practice, we extract a number of features in each image and use feature matching algorithms [13] to establish correspondence between the images. The resulting set of feature matches between two images usually have a subset of wrong ("outlier") matches due to errors in the feature extraction and matching process. We handle these outliers in a RANSAC framework [14] which attempts to find the motion parameters by first identifying the set of inlier feature matches. If we have an image sequence, we use feature tracking algorithms like KLT [15] to track a set of features through an image sequence. The correspondences specified by these tracks are used to compute the motion model parameters.

Usually, neither the scene being viewed is planar nor the motion a pure rotation. In

such cases, there is no linear map that relates one image to the other unless one neglects the effect of translation (similar to assumption made in (2.4)). In such cases, researchers either make simplifying assumptions based on the domain knowledge or include additional constraints involving more views to take care of the limitations of the geometric approach. In [16], Morimoto *et al.* demonstrate real time image stabilization that can handle large image displacements based on a two-dimensional multi-resolution technique. [17] propose an operation called *threading* that connects two consecutive Fundamental matrices using the trifocal tensor as the thread. This makes sure that consecutive camera matrices are consistent with the 3D scene without explicitly recovering it.

2.2 Related work

2.2.1 Video Stabilization

Video stabilization involves registering consecutive pairs of images of a sequence to each other using an appropriate motion model such as affine or homography. Image registration methods have widespread applications and the techniques can be broadly classified into intensity-based, flow-based and feature-based techniques.

In the intensity-based approach, the motion model parameters are solved by minimizing the sum of squared differences between the intensity values of corresponding pixels of the two images after warping using the geometric transform. The inverse-compositional alignment algorithm of Baker et. al. [18] is an efficient way to solve the optimization problem to perform the alignment. Bartoli [19] generalized this when the pair of images differ by a geometric and photometric transformation. In case the inten-

sity values of corresponding pixels between the two images are related by an unknown transformation (such as in multi-sensor registration), implicit similarity functions in the gradient domain have been used for registration purposes [20].

In the flow-based approach [21, 22], the optical flow between the pair of images is used for registration. The registration may be either a full-frame warp obtained by fitting a model to the flow, or a non-uniform warp using the dense motion vectors to take into account the varying displacements of parallax pixels.

The feature-based approach relies on abstracting an image as a set of salient feature points with associated geometric locations. Registration is performed by finding corresponding points between the images [23] and using this to estimate the motion model parameters.

Detecting moving objects in a video sequence involves stabilization followed by identifying the pixels that are in misregistration. Background subtraction [24] is one of the most commonly used techniques to identify moving objects. When two registered images are subtracted, the pixels belonging to the background have a small absolute value. However, the moving object pixels do not register using the background model and hence light up with high values of absolute differences. However, even slight misregistrations lead to false moving object detections which is a serious problem in the background subtraction framework.

Birchfield and Pundlik [25] propose a framework to combine local optic flow with global Horn-Schunck constraints [26] for joint tracking of features and edges. They express the idea of motion coherence in terms of a smoothness prior. The key difference of our work is that we incorporate motion coherence between features by solving a classifi-

cation problem to label features. The motion of features belonging to the same class are solved jointly by computing the model parameters.

Sheikh et. al. [27] propose a background subtraction framework where they track densely distributed points in a video and segment them into background and moving points. They use feature segmentations to infer the piecewise dense image segmentation using MRF priors. The proposed work is similar to [27] in that we also use the segmented set of features for pixelwise segmentation and moving target detection. The key difference is that the joint tracking and segmentation strategy enables us to track features uniformly distributed in the image plane (including homogeneous regions) making the segmentation step more reliable. The earlier work relies on accurately tracked individual feature points and this assumption does not hold in our scenarios. In addition, we use the foreground segmentations for target initialization and tracking.

Buchanan and Fitzgibbon [28] propose a feature tracking algorithm by combining local and global motion models. More accurate tracks are obtained by generating predictions of feature location using global models. Shi and Tomasi [15] proposed a method for feature monitoring in KLT tracking, by measuring feature dissimilarities after registration with an affine model. They use these dissimilarities as a criterion to judge the goodness of feature tracks. We use feature dissimilarities produced by the motion model as a measure of the likelihood of the feature belonging to the model. These dissimilarities are used to derive the probabilities of belonging to the common background model.

2.2.2 Video Mosaicking

Mosaicing is the process of compositing or piecing together successive frames of the stabilized image sequence so as to virtually increase the field-of-view of the camera [29]. This process is especially important for remote surveillance, tele-operation of unmanned vehicles, rapid browsing in large digital libraries, and in video compression. Mosaics are commonly defined only for scenes viewed by a pan/tilt camera, for which the images can be related by a projective transformation. However, recent studies have looked into qualitative representations, non-planar embeddings [30, 31] and layered models [32]. The newer techniques permit camera translation and gracefully handle the associated parallax. These techniques compute a "parallax image" [33] and warp the off-planar image pixels on the mosaic using the corresponding values in the parallax image. Mosaics represent the real world in 2D, on a plane or other manifold like the surface of a sphere or "pipe". Mosaics that are built on spherical or cylindrical surfaces belong to the class of panoramic mosaics [34, 35]. For general camera motion, there are techniques to construct a mosaic on an adaptive surface depending on the camera motion. Such mosaics, called manifold mosaics, are described in [31, 36]. Mosaics that are not true projections of the 3D world, yet present extended information on a plane are referred to as *qualitative* mosaics.

Several options are available while building a mosaic. A *simple* mosaic is obtained by compositing several views of a static 3D scene from the same view point and different view angles. Two alternatives exist, when the imaged scene has moving objects, or when there is camera translation. The *static* mosaic is generated by aligning successive images

with respect to the first frame of a batch, and performing a temporal filtering operation on the stack of aligned images. Typical filters are pixelwise mean or median over the batch of images, which have the effect of blurring out moving foreground objects. In addition, the edges in the mosaic are smoothed, and sharp features are lost. Alternatively the mosaic image can be populated with the first available information in the batch.

Unlike the static mosaic, the *dynamic* mosaic is not a batch operation. Successive images of a sequence are registered to either a fixed or a changing origin, referred to as the *backward* and *forward stabilized* mosaics respectively. At any time instant, the mosaic contains all the new information visible in the most recent input frame. The fixed coordinate system generated by a backward stabilized dynamic mosaic literally provides a snapshot into the transitive behavior of objects in the scene. This finds use in representing video sequences using still frames. The forward stabilized dynamic mosaic evolves over time, providing a view port with the latest past information supplementing the current image. This procedure is useful for generating an enlarged virtual field of view in the remote operation of unmanned vehicles.

In order to generate a mosaic, the global motion of the scene is first estimated. This information is then used to rewarp each incoming image to a chosen frame of reference. Rewarped frames are combined in a manner suitable to the end application.

2.2.3 Structure from Motion

Structure from Motion algorithms can be broadly classified into the following categories: batch techniques, minimal solutions and recursive frameworks. A comprehensive

survey may be found in [37].

In batch techniques, we jointly solve for the views of all the cameras and the structure of all the points. Bundle adjustment (BA) [38] is the representative algorithm in this class, and it minimizes an error function measuring the disparity in the detected image features and those generated from the current reconstruction. This is a non-linear least squares minimization problem that is commonly solved using the Levenberg-Marquardt (LM) [39] algorithm. The linear system corresponding to the normal equations for LM is intractable for large reconstruction problems. To handle large problems, the sparse structure of the Hessian matrix is used for efficiently solving the normal equations resulting in the Sparse Bundle Adjustment (SBA) [40] algorithm. The conjugate gradient (CG) method [39] is another choice for optimizing the reprojection error that does not require the solution of a large system; however suitable pre-conditioners are necessary for it to work well [41]. In addition, its benefits have not yet been clearly demonstrated for large scale problems.

Minimal solutions take a small subset of features as input and efficiently solve for the views of two or three cameras. A recent and popular algorithm in this class is the five-point algorithm which was proposed by Nister [42]. Because of the small number of points and views, solving for the parameters is very efficient. However, there may be multiple solutions and we need to disambiguate the various alternatives using additional points. An issue with this technique is the propagation of error while trying to stitch the various individual reconstructions together.

Recursive algorithms perform an online estimation of the state vector which is composed of the location and orientation of the camera, and the 3D locations of the world

landmarks. The estimation proceeds in a prediction-update cycle, where a motion model is used to predict the camera location at the next time step, and this estimate is updated using new observations. Simultaneus Localization and Mapping [43] methods also fall within the recursive framework since they employ a recursive filter to estimate the camera location in a causal fashion.

Another class of techniques is factorization-based approaches [12, 44] that take the measurement matrix (containing the feature points observed in all views) and factorize it into a product of the motion and structure matrices.

Although solving for 3D locations of points from feature tracks is important, it is just one part of the problem. There are other open research problems in generating texture-mapped 3D models of environments after solving for SfM from image sequences. Researchers have been actively working on ways to perform 3D model acquisition from ground-based sequences. Fruh and Zakhor [45] describe an automatic method for fast, ground-based acquisition of 3D city models using cameras and laser scanners. In [46], they describe a method for merging ground-based and aerial images for generating 3D models. Akbarzadeh et. al. [47] introduce a system for automatic, geo-registered 3D reconstruction of urban scenes from video.

In spite of this large body of work in SfM, it is a very hard ill-posed and inverse problem and very few of these algorithms provide satisfactory performance in real-world scenarios. The main difficulties faced by these algorithms in real world scenarios are in establishing correspondence across image sequences, feature tracking errors, mismatched correspondences, occlusion etc.

Therefore, recent research has focussed on developing SfM algorithms in the pres-

ence of additional constraints on the problem. For instance, position information about the cameras from Global Positioning Systems (GPS) can help us solve for the parameters [48]. If inertial measurements from IMUs are available, we can reduce the ambiguities in the SfM problem [49].

Our work is related to prior work in SfM literature that assume a visible dominant plane in the scene. An important algorithm in this class is the use of the plane-plus-parallax model for the recovery of 3D depth maps [50]. The multi-view constraints imposed by a plane were used by Rother [51] and Kaucic [52] to simplify the projective reconstruction problem into a linear system of equations. Bartoli [53] derived a linear algorithm for estimating the structure of objects moving on a plane in straight lines without rotating. Reconstruction of objects moving in an unconstrained fashion was studied in detail by Fitzgibbon and Zisserman [54].

A special case of our algorithm is Structure from Planar Motion (SfPM) where we have a surveillance scenario with a static camera observing moving objects on a plane. In this case, the measurement matrix simplifies into a product of a motion and a structure matrix that is of rank 3. Li and Chellappa [55] describe a factorization algorithm for solving for the structure and planar motion.

Another special case, leading to a linear multiview reconstruction, arises when we know the homographies between successive images induced by the planar scene. Rother [51] stabilizes the images using homographies, and chooses a projective basis where the problem becomes one of computing structure and motion of calibrated translating cameras. They derive linear equations for the camera centers and points, and simultaneously solve the resulting linear system for all cameras and points using SVD based

techniques. The performance of their algorithm is heavily dependent on the estimate of the homographies. In addition, the memory and computational requirements of the algorithm become infeasible when we have a large number of frames and points.

Our algorithm assumes approximate knowledge of a certain direction vector in each image of the sequence and also the altitude from a plane perpendicular to this vector. These quantities are well defined when we observe a dominant plane in the scene, but the algorithm does not require the visibility of a plane (for stationary scenes). In this respect, it is quite different from all other previous approaches.

The proposed algorithm belongs to the class of alternation algorithms that solve for the structure and motion in an iterative fashion [44, 56]. These approaches solve for the projective depths along with the motion and structure matrices and result in a projective reconstruction. The projective depths are typically initialized to unity and later refined iteratively. This has been reported to work well only in special settings where the depth of each feature point remains approximately the same throughout the sequence [37]. This does not cover many important scenarios such as roadside urban sequences or aerial videos where the altitude of the camera varies a lot. Our algorithm makes use of the bilinear form in the Euclidean frame without slack variables. Hence it does not have any restrictions on its use except that the gravity and height measurements must be available.

Oliensis and Hartley [57] published a critique on the factorization-like algorithms suggested in [44, 56] for projective SfM. They investigated the theoretical basis for performing the suggested iterations in order to decrease the error function. They analyzed the stability and convergence of the algorithms and concluded theoretically and experimentally that the Sturm-Triggs [44] as well as the slightly modified Mahamud [56] algorithm

converges to trivial solutions. They analyzed the error functions for these algorithms and showed that the stationary points were either local minima that corresponded to trivial solutions or they were saddle points of the error. Based on this analysis, we investigate the stability and convergence properties of the proposed algorithm.

Chapter 3

Intensity-based Video Stabilization

The intensity-based registration algorithm of Baker et. al. [18], known as simultaneous inverse compositional alignment, iteratively minimizes the sum-squared difference of pixel intensities between corresponding pixels of a pair of images. We discuss the application of this algorithm for the registration of low-quality and low-resolution video obtained from aerial platforms [3].

3.1 Intensity-based alignment

In intensity-based registration techniques, we assume a global parametric transformation that registers the two images. We pose the registration task as an optimization problem involving the minimization of the intensity difference between the two images registered using the current parameters. Suppose I_1 and I_2 are the two images to be registered. We solve the following minimization problem:

$$\min_{p} \sum_{q \in R} \|I_1(q) - I_2(P(q;p))\|^2 \tag{3.1}$$

where q denotes the coordinates of a pixel, p denotes the transformation parameters of the registration model, $P(q;p)$ denotes the global transformation applied to the pixel q, and R is the region of I_1 over which the image intensity differences are computed.

Common choices of the motion model $P(q;p)$ used for registration are the affine model, homography etc. The region R over which the intensity differences are computed

is typically chosen to be the entire region of overlap between the registered images. The minimization is performed using an iterative minimization technique that linearizes the objective function (using a Gauss-Newton approximation) at each iteration and computes an update that minimizes the error. A pyramidal implementation of a Gauss-Newton minimization procedure was introduced by Bergen et. al. [58]. More recently, Baker et. al. proposed a fast "inverse compositional" algorithm [18] for minimizing (3.1).

We sketch the details of the inverse compositional algorithm. Let $P(q;p)$ denote the current transformation between the images. Let $\Delta P(q;\delta_p)$ denote an update to the current transformation that reduces the image-difference error. We apply the update to the transformation by composing the current transformation with the inverse of the incremental transformation. The update rule is $P(g;p) \leftarrow P(\Delta P^{-1}(q;\delta_p);p)$.

The optimization (3.1) is performed over δ_p, the parameter of the incremental transformation, and can be recast as follows:

$$\min_{\delta_p} \sum_{q \in R} \left\| I_1(q) - I_2(P(\Delta P^{-1}(q;\delta_p);p)) \right\|^2 \tag{3.2}$$

The inverse compositional trick is to apply the incremental transformation to the first image I_1 as opposed to the second image. The problem can be rewritten as:

$$\min_{\delta_p} \sum_{q \in R} \left\| I_1(\Delta P(q;\delta_p)) - I_2(P(q;p)) \right\|^2 \tag{3.3}$$

We linearize this equation using a Taylor series expansion of the intensities of I_1 to obtain:

$$\min_{\delta_p} \sum_{q \in R} \left\| I_1(q) + \nabla I_1^T(q) \nabla \Delta P(q;\delta_p)) - I_2(P(q;p)) \right\|^2 \tag{3.4}$$

This technique is applicable only when there is a correlation in the intensities of corresponding pixels between the two images. Extensions of the inverse compositional al-

gorithm have been proposed [19] when the intensities of corresponding pixels are not equal, but there is a relation between the intensities such as a gain and a bias, or an affine transformation.

3.1.1 Effect of region of registration

Baker's algorithm is traditionally applied to minimize the sum-of-squared intensity difference of all the corresponding pixels in the image pair. In other words, the set R denoted above is chosen to consist of all the pixels of the image. We argue that this most common default choice does not necessarily result in better registrations. We point out that using a subset of the image pixels for registration can result in better registration results in lesser computation time. The following are two scenarios where registration may be defined over a subset of pixels.

1. *Multisensor image registration:* Images taken from different sensors have no correlation between their pixel intensities. Multisensor registration techniques use implicit similarity functions that are defined for pixels marking prominent features or with high gradient magnitudes.

2. *Low-quality and low-resolution video:* In videos obtained from airborne platforms with small cameras, the noise level in intensities is high under low illumination conditions. In these conditions, image motion is perceived due to the motion of pixels with high gradients defining prominent edges. This is because edges are preserved under high image noise levels. Since pixels with small gradients do not contribute towards perception of motion, perceptually better registration results may be ob-

tained by using only pixels with high image gradients for intensity-based registration.

We apply this concept to the registration of images in a video sequence, where the noise levels in the images are high. We demonstrate that in this case, we improve the convergence of the inverse compositional alignment algorithm by using a small subset of image pixels for registration.

3.1.2 Experiments

We select a set of 10 representative images from a video sequence captured from a low-resolution camera onboard an aerial platform. The chosen images contain large regions with very little texture information. In other words, the image gradients are very small within these regions and hence these regions do not contribute towards the measurement of image motion. Figure 3.5(a) illustrates a sample image from the dataset.

For each image, we estimate a homography transformation (by displacing four chosen points by a pre-specified amount), and warp the images according to the homography. We test the convergence properties of the registration algorithm by registering the original and warped image, and measuring the convergence rate, number of iterations, etc. We perform these experiments for various levels of distortion and various levels of Gaussian noise added to one of the images. In applying the intensity-based algorithm, we first compute the gradient magnitudes of the first image at each pixel, and then order the pixels according to the gradient magnitudes. We select a chosen fraction of pixels (such as 25%) which have the highest gradient magnitudes and then use these pixels for registration. To

preserve the gradient information, we also select surrounding pixels within a small window. An illustration of the pixels selected from the image in figure 3.5(a) for use in one run of the intensity-difference minimization is shown in Figure 3.5(b).

Figure 3.1 plots the standard deviation of image pixel noise versus the number of iterations taken for convergence. The figure shows five different curves corresponding to the noise versus iterations plots when different percentages of pixels were used for registration $(100\%, 83\%, 62\%, 41\%, 24\%)$. We observe a well-defined variation in the number-of-iteration plots for various saliency ratios. As the saliency ratio decreases (implying that we use fewer pixels for registration), the number of iterations needed for convergence goes down. We also experimentally verified that the time-per-iteration of registration goes down linearly with the number of pixels used.

Figure 3.2 plots the standard deviation of the image pixel noise versus the geometric error of registration. The geometric error is defined as the ground-truth registration parameters and the solved parameters. The distance is measured as an $L2$ norm of the difference between the two parameter vectors. From the plot, we observe that there is little difference between the geometric registration error for the various registration trials carried out at different saliency ratios. Unlike the plots for the number of iterations, we do not observe a specific variation in the geometric error curves for different saliency ratios. However, notice that for several noise levels, the geometric error obtained by registration with the lowest saliency ratio is actually higher than the geometric error obtained with the highest saliency ratio. These observations imply that we can do better registration of the two images, by using fewer iterations (and lower computation time), by choosing a subset of image pixels (salient pixels).

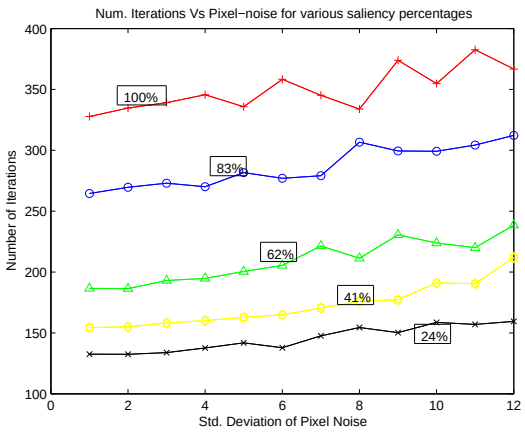


Figure 3.1: A plot of the number of iterations taken for convergence versus the standard deviation of image pixel noise for different fractions of number of pixels used in the registration. As the number of salient pixels used for registration decreases, the number of iterations used for convergence decreases too.

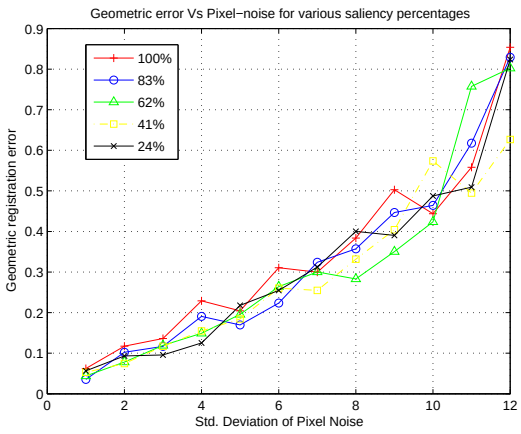


Figure 3.2: A plot of the geometric error versus the standard deviation of image pixel noise for various ratios of salient pixels used in registration. For various saliency rations, accuracy of registration is almost similar. In some cases the accuracy of registration is greater for lesser number of pixels used.

3.2 Video Mosaicking

Constructing mosaics out of video sequences is an important application for cameras deployed on MAVs and UAVs, where the sensors are of low-resolution ("strawhole sensors"). Building mosaics of these sequences will give an enhanced field-of-view of the scene for an operator viewing the video. The noise-level in the images due to the low camera quality needs to be addressed while mosaicing video sequences. Because of the changing noise-level, the error in the estimates of the parameters of the image motion model varies for different images. During the process of developing the mosaics, only those frames must be stitched whose image motion parameters lead to good registration of the images.

The following criterion can be used as a metric for measuring the registration qual-

ity [3].

$$R(I_M, I_t) = D(I_M, I_t) + G(I_M, I_t)$$

$$D(I_M, I_t) = \sum_{r \in R} [I_M(r) - I_t(p(r; m))]^2$$

$$G(I_M, I_t) = \sum_{r \in R} [\nabla I_M(r) - \nabla I_t(p(r; m))]^2 \tag{3.5}$$

The above criterion (3.5) $R(\cdot, \cdot)$ depends on the image difference error $D(\cdot, \cdot)$ as well as the image-gradient difference error $G(\cdot, \cdot)$. I_t is the current frame and I_M is the current mosaic image. The region R denotes the region of overlap between I_t and I_M. The second error measure $G(I_M, I_t)$ is derived from gradient-domain image registration methods [59, 60, 20, 61]. It primarily measures the mis-registration of high-gradient pixels in both the images. The reason for adding this extra term is because in low quality images, the image difference error (that works on raw intensities) by itself does not accurately reflect the mis-registration between two images. The gradient error term (applied on smoothed images) measures the mis-registration between prominent edges in the image.

We start off with an empty mosaic image, and sequentially fill the pixels in the mosaic with the information in each frame. The error measure (3.5) is computed for each frame and the mosaic is updated with information from a frame only if its registration error is below a chosen threshold. There are two ways in which the pixels of the current frame are used to update the mosaic: either as reference frame pixels, or as non-reference frame pixels. Reference frames are directly pasted onto the mosaic whereas non-reference frames update only unfilled pixels on the mosaic.

- If the most recent reference frame is more than k frames away from the current frame in the sequence (for an appropriately chosen k), then the current frame is

incorporated as a reference frame in the current frame as a reference frame.

- If the overlap between the current frame warped and the last reference frame (when warped on the mosaic) falls below a threshold, we incorporate the current frame as a reference frame.

- If neither of the above two conditions are satisfied, the current frame is incorporated as a non-reference frame.

3.2.1 Results

We present some image mosaicks which are the results of the proposed registration and mosaicking algorithm when it is applied on a video sequence captured using a video camera onboard an MAV. The air-vehicle is flying in a jittery fashion and the video resolution (230×310) is small. In addition, there is a lot of motion between consecutive frames of the sequence. In figs. 3.3 and 3.4, we show mosaics of two sequences that we generated using our algorithms. Figure 3.5(a) illustrates an image of the sequence and 3.5(b) illustrates the set of pixels in that image used for registration purposes.

Figure 3.3: This figure shows a mosaic of a sequence obtained by registering the images of a sequence using the intensity-based stabilization algorithm. Approximately 30% of the pixels in each image were used for the purpose of registration.

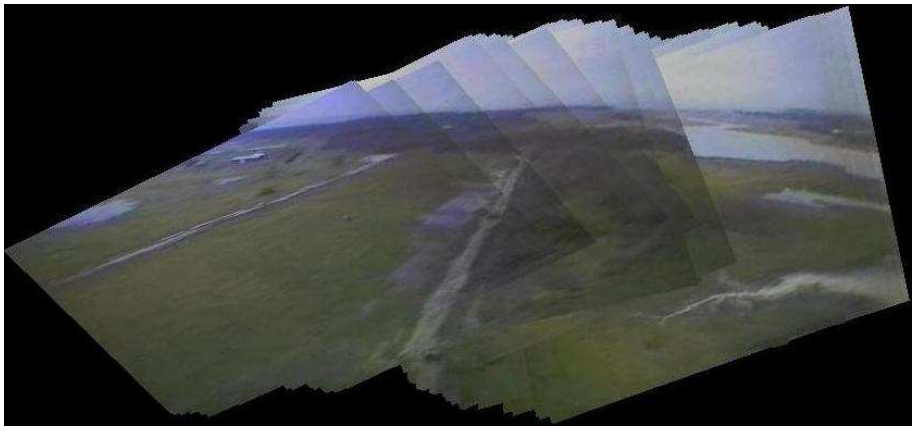
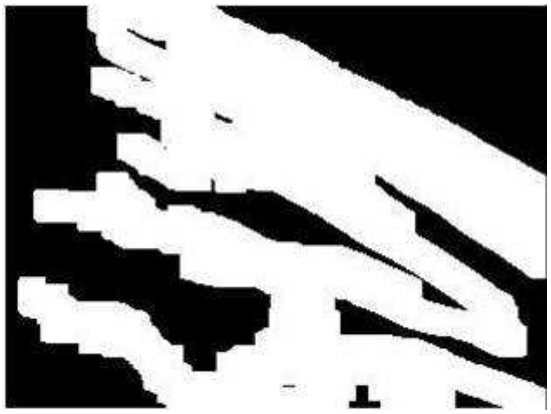


Figure 3.4: This figure shows a mosaic of another sequence obtained by registering the images using the intensity-based stabilization algorithm. Approximately 30% of the pixels in each image were used for the purpose of registration.

(a) Sample Image



(b) Selected regions for minimization shown in white

Figure 3.5: Figure 3.5(a) shows a sample image of a sequence captured from an aerial platform. The resolution is 230×310 pixels. We use our algorithm to build a mosaic of the sequence. The regions of the image selected by our algorithm for minimizing the image difference error with the mosaic is shown in Figure 3.5(b). Note that these regions contribute the most to our perception of image motion.

Chapter 4

Feature-based Stabilization and Motion Segmentation

4.1 Feature-based Image Registration

Intensity-based registration techniques described in the last chapter are useful for registering images when we have prior information about the subset of pixels that belong to the dominant motion model. If the image sequence contains moving objects, we do not know the moving object pixels prior to stabilization. Any measures for selecting a salient set of pixels in one image may select pixels belonging to multiple motion layers. Solving for the registration parameters using a pixel set containing outlier pixels not belonging to the motion model may lead to biased motion parameter estimates. In order to handle moving objects in the scene, we need to segment out the different motion layers in the video along with solving for the parameters of the dominant motion model. A feature-based representation of video is more convenient for the problem of registration and segmentation.

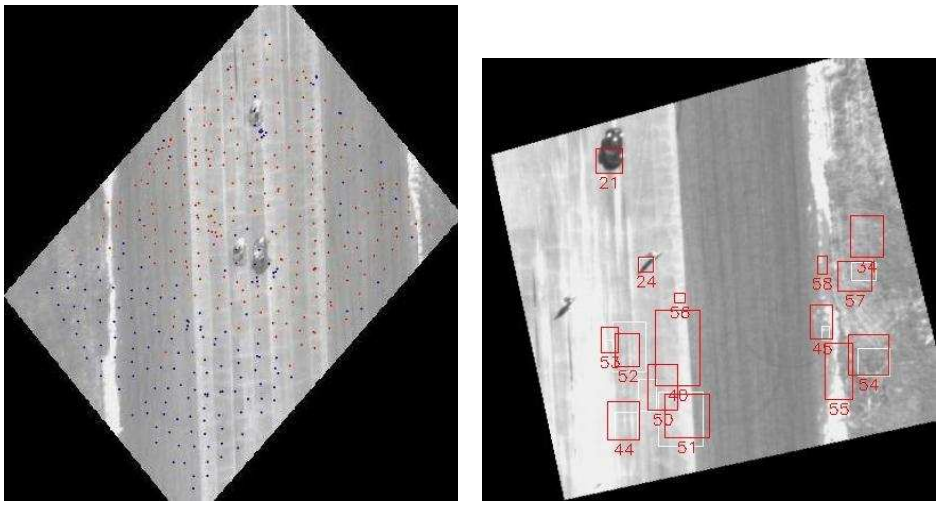
4.1.1 Shortcomings of KLT tracking and background subtraction

Traditionally, features are tracked through the image sequence and RANSAC [14] is used to identify the inlier features belonging to the dominant background motion model. However, tracking errors lead to many misclassifications of features. Figure 4.1(a) illustrates an example of an image from our aerial video dataset with KLT [62] tracked

features overlaid on it. The image has very few prominent features on the background, and there are large regions of homogeneous texture. In addition, the road in the middle contains repeated patterns making it difficult to solve for the correct displacement. These tracking errors result in a large number of background features to be classified as outliers by a model selection algorithm such as RANSAC.

When the video is stabilized using the solved background displacements and a background subtraction method is employed to detect moving objects, there are many spurious motion blobs due to slight misregistrations at the edges. Figure 4.1(b) illustrates an image with detected moving objects overlaid on top. Many of the boxes are false alarms enclosing false motion blobs near edges. Some of these false alarms may be reduced by temporal filtering or choosing a higher threshold for motion detection. However, this would fail to detect the small moving objects in the sequence.

We propose an algorithm for simultaneous registration and moving object detection that uses feature tracking but comes up with a dense motion segmentation of the scene [63]. We assume a full-frame parametric background motion model and directly solve for the model parameters using all the features belonging to the model. Since multiple features are used for solving the model parameters, this enables us to select a large number of features, densely and uniformly distributed in the image plane, and use the global motion constraint to accurately track the features. We will now describe the joint tracking and segmentation algorithm.



(a) (b)

Figure 4.1: This figure illustrates the failure of KLT feature tracking method. Figure 4.1(a) shows an image with overlaid features. The red points are classified to be on the background and blue points are classified to be on the foreground. Ideally, blue points must be restricted to moving vehicle features. Note the large number of blue points on the background indicating misclassifications. Figure 4.1(b) shows an image with boxes overlaid on top, indicating detected moving blobs after a background subtraction algorithm was applied on registered images. Notice the number of falsely detected moving objects. The underlying video sequence was very well stabilized upon evaluation by a human observer, but outlier pixels near edges show up as moving pixels in background subtraction.

4.1.2 Joint tracking

Consider two consecutive images I and J in a sequence. Assume that the image motion comprises of a pure translational motion along the X and Y axes. Assume that N salient feature points $(f_1, f_2, \cdots, f_N)$ have been detected in the image I that need to be tracked on image J. Let the image displacements be described by d which is a two-tuple denoting the translations along the x and y directions of the image. The objective is to solve for d by jointly tracking the set of n feature points. We solve the following optimization problem.

$$\arg\min_{d} \sum_{i=1}^{N} \int\int_{x \in W(f_i)} \left[J(x + \frac{1}{2}d) - I(x - \frac{1}{2}d) \right]^2 \cdot w(x - f_i) dx \tag{4.1}$$

In this optimization, the double integral denotes the accumulation of the objective function over a small window around the feature. The term

$$\int\int_{x \in W(f_i)} \left[J(x + \frac{1}{2}d) - I(x - \frac{1}{2}d) \right]^2 \cdot w(x - f_i) dx \tag{4.2}$$

is the same as the objective function that is minimized in the Lucas-Kanade feature tracking method [62], where one solves for the displacement of each feature patch separately, and independently of other features. In our case, the difference is that we solve for a common displacement of a set of features that belong to the same motion model, by summing the image patch difference errors for each feature. We solve (4.1) by expressing the image intensities I and J at a displaced pixel point in a Taylor series around the neighboring pixel locations. The optimization (4.1) reduces to:

$$\arg\min_{d} \sum_{i} \int\int_{x \in W(f_i)} \left[J(x) - I(x) + g^T d \right]^2 \cdot w(x - f_i) dx \tag{4.3}$$

where

$$g = \begin{bmatrix} \frac{\delta}{\delta x}\left(\frac{I+J}{2}\right) & \frac{\delta}{\delta y}\left(\frac{I+J}{2}\right) \end{bmatrix}^T \tag{4.4}$$

We differentiate (4.3) with respect to d to get

$$\sum_i \int\int_{x \in W(f_i)} g \cdot \left[J(x) - I(x) + g^T d\right] \cdot w(x - f_i) dx = 0 \tag{4.5}$$

which simplifies to

$$\left[\sum_i \int\int_{x \in W(f_i)} g(x) \cdot g(x)^T w(x - f_i) dx\right] d = \sum_i \int\int_{x \in W(f_i)} \left[J(x) - I(x)\right] g(x) w(x - f_i) dx \tag{4.6}$$

We solve the above linear system to obtain the solution for d as follows.

$$d = \left[\sum_i \int\int_{x \in W(f_i)} g(x) \cdot g(x)^T w(x - f_i) dx\right]^{-1} \cdot \sum_i \int\int_{x \in W(f_i)} \left[J(x) - I(x)\right] g(x) w(x - f_i) dx \tag{4.7}$$

This derivation is applicable when the image motion is described by a pure translation model. A similar set of equations can be derived for more general models such as affine motion. We assume a translation model for simplicity of derivation.

4.1.3 Robustness

In the joint tracking framework described above, the presence of outliers in the pool of features can bias the translation estimates. The quadratic error function measuring the sum squared differences of image patches is very sensitive to the presence of outliers that do not follow background motion. To address this, we introduce robust metrics in the optimization criterion (4.3) to make the solution robust to the presence of outliers. We

use the Huber's robust loss function defined as follows.

$$\rho(u) = \begin{cases} \frac{1}{2}u^2 & \text{if } |u| \leq c \\ c(|u| - \frac{c}{2}) & \text{if } |u| > c \end{cases} \tag{4.8}$$

In robust joint-tracking the sum-of-squares objective function in (4.9) is modified using the Huber loss function (4.8) to obtain the following robust optimization problem.

$$\arg\min_{d} \sum_{i} \int\int_{x \in W(f_i)} \rho \left(J(x + \frac{1}{2}d) - I(x - \frac{1}{2}d) \right) \cdot w(x - f_i) dx \tag{4.9}$$

With the introduction of the robust loss-function, the optimization problem (4.9) can be solved using gradient descent techniques such as the conjugate gradient method. The advantage of introducing the robust metric is that the solution is much more resilient to the presence of outliers in the pool of features used for joint tracking. We evaluate the performance of joint tracking and the robustness to outliers introduced by the Huber loss function. We choose a set of set of 20 images from the dataset. For each image, we generate a second shifted image by translating the first image in the Y direction. Then, we simulate three moving image patches that move differently from the background. This procedure effectively produces two images that are shifted versions of each other except for the moving image patches that do not follow the background motion. We select features on the background and the moving patches and solve for the displacement jointly. For the purposes of joint tracking, all features selected on the moving patches are outliers. For different fractions of outliers in the feature pool, we compare the joint tracking solution with the robust joint tracking solution. Figure 4.2 plots the solved displacements for various fractions of outliers. The blue line indicates the ground truth displacement, and the red curve indicates the averaged displacement obtained from joint tracking. It is

clear that least-squares tracking is highly sensitive to outliers. The green curve plots the average displacement with robust tracking, and it is much more robust to outliers. With 60% inliers in the feature pool, the solved displacement is very close to the individual displacement.

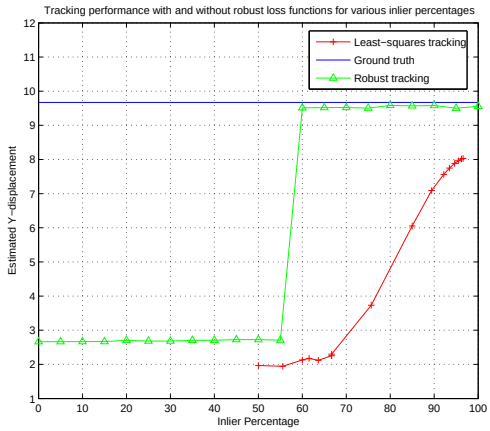


Figure 4.2: This figure plots the estimated Y-displacement between two images estimated by jointly tracking features with and without using robust loss functions. The red-curve plots the estimated displacement using a sum-of-squares cost function (similar to traditional KLT). The blue line shows the ground-truth displacement and the green-curve shows the estimated displacement obtained by using robust loss-functions.

4.1.4 Segmentation

To estimate the background motion parameters by tracking features jointly, the segmentation of features into background and moving features is necessary. However, since class labels are unknown, the segmentation problem must be solved along with background parameters. In a joint background tracking framework, a better segmentation enables more accurate joint feature tracking, and accurate background model parameters give rise to a better segmentation. We use a simultaneous tracking and labeling approach to solve for the background motion parameters and feature class labels. We have a two-

class classification problem for each feature point.

H_0 : feature f_i belongs to the background motion model

versus

H_1 : feature f_i does not belong to background motion model

Consider a video sequence consisting of F frames denoted by $(I_1, I_2, \cdots, I_F)$. Assume that N features are tracked through the sequence. Let the i^{th} feature in the frame I_t be denoted by $f_{i,t}$. Let the background model displacement between frame I_t and I_{t+1} be denoted by $\vec{d}_t$. We maintain for each feature the probability that it belongs to each model. Let $p(f_i \in H_0 \mid \vec{d}_{t-\alpha+1:t})$ denote the probability of feature i belonging to the background given the background model displacements for α frames in the past. Then, $p(f_i \in H_0 \mid \vec{d}_{t-\alpha+1:t}) = 1 - p(f_i \in H_1 \mid \vec{d}_{t-\alpha+1:t})$. α is the length of a suitably chosen temporal sliding window to evaluate the probability based on motion in the past α frames. The length of the sliding window is chosen to balance out the following opposing factors: (1) Using multiple frames in the past leads to more accurate estimation of probability by protecting against possible mis-estimation in the last frame. (2) Using too many frames is not suggested as it does not capture changing feature identities.

The Bayes theorem is used to calculate the probability of a feature belonging to the background given all the evidences of background motion parameters from the past frames in the feature window.

$$p(f_i \in H_0 \mid \vec{d}_{t-\alpha+1:t}) = \frac{p(f_i \in H_0) \cdot p(\vec{d}_{t-\alpha+1:t} \mid f_i \in H_0)}{p(f_i \in H_0) \cdot p(\vec{d}_{t-\alpha+1:t} \mid f_i \in H_0) + p(f_i \in H_1) \cdot p(\vec{d}_{t-\alpha+1:t} \mid f_i \in H_1)} \tag{4.10}$$

We assume that the background motion between consecutive frames is independent; hence they are independently estimated.

$$p(\vec{d}_{t-\alpha+1:t} \mid f_i \in H_0) = \prod_{\tau=t-\alpha+1}^{t} p(\vec{d}_\tau \mid f_i \in H_0) \tag{4.11}$$

4.1.5 Joint Tracking and Segmentation Algorithm

Given the background motion and feature membership probabilities up to frame t, we estimate the background motion at frame $t+1$ (between I_{t+1} and I_{t+2}), and re-estimate the feature membership probabilities using the new frames. We iteratively switch between background model estimation and membership probability re-estimation steps.

4.1.5.1 Motion estimation

Let $d_{t+1}^{(k)}$ denote the background motion parameters between I_{t+1} and I_{t+2} at the k^{th} iteration. We rewrite the joint tracking objective function by weighing the contribution from each feature by the corresponding membership probability. We estimate the displacement $d_{t+1}^{(k+1)}$ by solving the modified robust optimization problem as follows.

$$\arg\min_{d_{t+1}^{(k+1)}} \sum_i \int\int_{x \in W(f_i)} p(f_i \in H_0 \mid \vec{d}_{t-\alpha+2:t}, \vec{d}_{t+1}^{(k)}) \cdot \rho\left(J(x + \frac{1}{2}d_{t+1}^{(k+1)}) - I(x - \frac{1}{2}d_{t+1}^{(k+1)})\right) \cdot w(x - f_i)dx \tag{4.12}$$

4.1.5.2 Membership probability estimation

With an estimate of the motion parameters from the motion estimation iteration, we update the feature membership probability $p(f_i \in H_0 \mid \vec{d}_{t-\alpha+2:t+1}) = p(f_i \in H_0 \mid \vec{d}_{t-\alpha+2:t}, \vec{d}_{t+1}^{(k+1)})$. This probability is computed using the Bayes rule in (4.10) and (4.11).

The term $p(d_t \mid f_i \in H_0)$ measures the probability that the background displacement at frame I_t is d_t using the single feature f_i which is known to belong to the background. We define this to be the probability that the displacement of feature location $f_{i,t}$ from I_t to I_{t+1} is d_t. This is computed based on the intuition that if d_t is the true background displacement, then the feature dissimilarity of $f_{i,t}$, computed as $\int\int_{x \in W(f_i)} (I_{t+1}(x + d_t) - I_t(x)) \cdot w(x - f_i)dx$, is low. On the other hand, if d_t is wrongly estimated, then the computed dissimilarity is high. Therefore, $p(d_t \mid f_i \in H_0)$ is related to the probability of observing the feature dissimilarity produced by d_t given that the feature f_i was an inlier.

We use a data-driven approach to find the distribution of feature dissimilarities. We track KLT features individually through a small subset of the sequence and accumulate the dissimilarities of the successfully tracked features. We compute the distribution of the feature residues using histogram estimation techniques. Then we fit several different parametric distributions to this histogram and find the best fitting distribution using statistical tests. We found that the dissimilarities were distributed according to a three parameter log-normal distribution as given below.

$$p(x) = \begin{cases} \frac{h}{\sigma\sqrt{2\pi}(x-\theta)} \exp(-\frac{(\log(x-\theta)-\zeta)^2}{2\sigma^2}) & x > \theta \\ 0 & x \leq \theta \end{cases} \tag{4.13}$$

where θ is the threshold, σ is the shape parameter which is a positive real number, and ζ is the scale parameter which is a real number.

If the feature f_i does not belong to the background model, then the background model parameter d_t leads to a relatively high residue for the feature if d_t is the correct background displacement. The term $p(d_t \mid f_i \in H_1)$ measures the probability that d_t is the background displacement given a feature f_i that does not belong to the background.

We define this to be the distribution of dissimilarities of features that are mistracked. We start with successfully tracked KLT features and then compute the dissimilarities after adding nominal errors to the solved displacements. We added between 2 to 3 pixels to the displacement which represented the difference between the background displacement and actual displacement of an outlier feature. We use the set of dissimilarities to find the distribution using histogram estimation techniques. We found that the log-normal distribution was a good fit in this case also. Figure 4.3 plots the inlier and outlier dissimilarity distributions obtained in this empirical fashion.

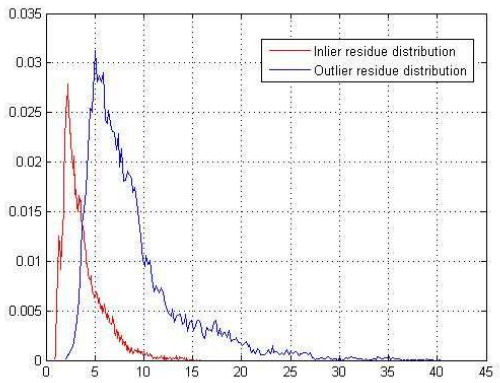


Figure 4.3: This figure shows the distribution of feature dissimilarities produced by the background model parameters for features lying on the background (inliers) and those on moving objects (outliers). The red curve plots the distribution of dissimilarity for inliers and the blue curve plots the distribution for outliers. These distributions are obtained using the data-driven approach described in the text, and are empirically found to be closest to the lognormal distribution.

4.1.6 Initialization and Implementation

We initialize the algorithm by selecting a large number of features in the first image and assigning them to the background model. We relax the feature selection thresholds of KLT by allowing features even with very poor conditioning to get selected. This has the

result of placing features inside homogenous regions along with locations in the image with strong gradients. Since we track the background features jointly using the idea of background motion coherence, the well-conditioned feature windows aid the motion estimation of poorly conditioned feature windows. This has the effect of producing feature tracks that are uniformly distributed in the image which helps us in the dense segmentation step.

We iterate the joint tracking and segmentation steps until convergence for each successive pair of images. Features whose probability of belonging to the background drops below 0.5 are labeled to be on the moving objects. The foregrond features are tracked individually using the traditional KLT equations. All the background features are assigned the parameters of the jointly solved background model.

In practice, we have observed that the mislabeled features in the first frame (during the initialization step) are quickly removed by the algorithm because their probability of belonging to the background quickly drops below 0.5. New features are initialized with equal probability of being in the background or foreground. As the algorithm proceeds through successive pairs of frames, these features are labeled very accurately.

The algorithm provides us with a set of feature tracks through the image sequence that are accurately labeled into foreground and background. These features are almost uniformly distributed on the image irrespective of the presence or absence of good features to track.

4.1.7 Dense Segmentation

The joint tracking and segmentation described above gives us a set of feature tracks distributed over the entire image area that is classified with high accuracy into features on the background and features on the moving objects. The objective of motion segmentation is to obtain a binary labeling of the image into background and foreground regions which involves labeling all pixels in the image. We use a MAP-MRF framework proposed in [64, 27] to infer the class labels of all the pixels from the labels of feature points. We summarize this algorithm here, as proposed in [64, 27], for the sake of completeness.

Assume there are N pixels in the image indexed by $i = \{1, \cdots, N\}$. Let the pixel labeling be denoted by $L = [l_1, l_2, \cdots, l_N]$ where $l_i \in \{0, 1\}$ denotes the label for pixel i with 0 denoting background and i denoting foreground. Let the feature locations in the image be represented by $X = [x_1, x_2, \cdots, x_F]$ where x_j corresponds to the location of the j^{th} feature and F denotes the number of features. We want to estimate

$$L^* = \arg\max_L p(L \mid X) \tag{4.14}$$

Using Bayes theorem, we can factor $p(L \mid X) \propto p(L)p(X \mid L)$. $p(X \mid L)$ can be split using conditional independence as $p(X \mid L) = \prod_{j=1}^{F} p(x_j \mid l_j)$. Hence, we can rewrite the likelihood as

$$p(L \mid X) \propto p(L) \prod_{j=1}^{F} p(x_j \mid l_j) \tag{4.15}$$

We need to define $p(x_j \mid l_j)$ appropriately. This is done in [27] by learning a model for background and foreground pixels using the labeled features. Assuming that ψ_b represents the background model and ψ_f represents the foreground model, the probability of the

feature point x_j given its class label is written as

$$p(x_j \mid l_j) = \begin{cases} p(x_j \mid \psi_b) & l_j = 0 \\ p(x_j \mid \psi_f) & l_j = 1 \end{cases} \tag{4.16}$$

The models ψ_b and ψ_f are expressed as non-parametric densities in the joint color and location (x, y, r, g, b) space. We use the Gaussian kernel in our experiments.

The smoothness prior is expressed in terms of an MRF that penalizes neighboring pixels that have different labels.

$$p(L) \propto \exp\left(\lambda \sum_{i=1}^{N} \sum_{j \in N(i)} (l_i l_j + (1 - l_i)(1 - l_j))\right) \tag{4.17}$$

The likelihood can be written as

$$p(L \mid x) = \lambda \sum_{i=1}^{N} \sum_{j \in N(i)} (l_i l_j + (1-l_i)(1-l_j)) + \sum_{i=1}^{N} l_i \log p(x_j \mid \psi_f) + \sum_{i=1}^{N} (1-l_i) \log p(x_j \mid \psi_b) \tag{4.18}$$

The optimal solution is efficiently computed using the graph-cuts algorithm [65, 66, 67].

4.2 Experiments

We present stabilization and moving object detection results on aerial sequences from the VIdeo Verification and IDentification (VIVID) dataset. The dataset has associated metadata with telemetry measurements providing information about the rotation of the camera.

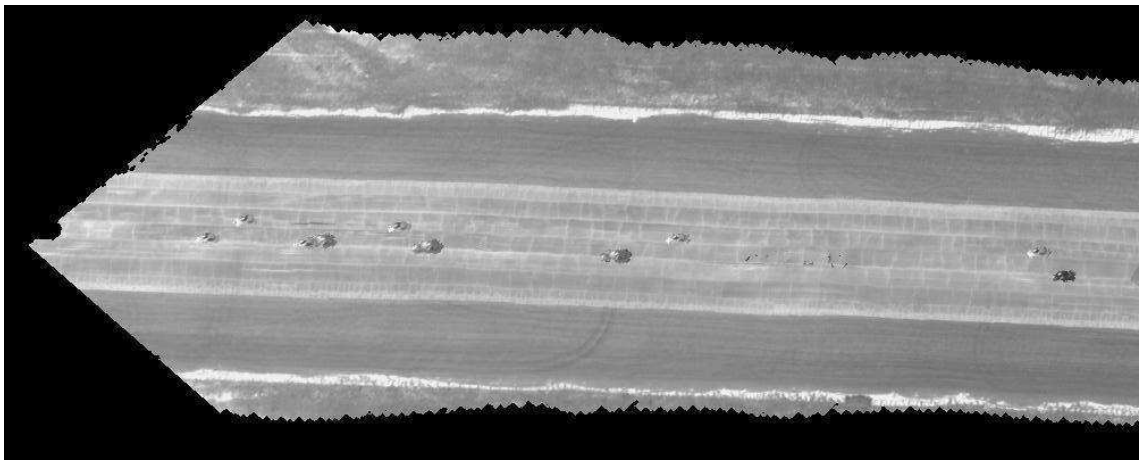
4.2.1 Metadata

The aerial platform used to capture the images consists of a gimbal mounted with the cameras. The gimbal provides measurements of the following quantities: platform

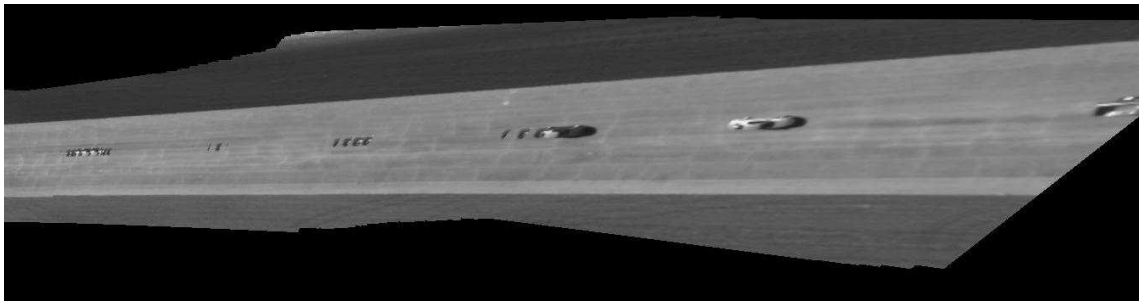
roll, pitch and heading, latitude, longitude and altitude, and sensor azimuth, elevation and twist. Using these measurements, we can compute the position and orientation of the camera in terms of a rotation matrix R and translation vector T with respect to the world coordinate system [68]. The 3×4 projection matrix between the image plane and world plane is written as $P = K \cdot R \cdot \begin{bmatrix} I & -T \end{bmatrix}$, where K is the camera calibration matrix and I is the 3×3 identity matrix. This projection matrix is used to transform each image of the sequence into a fronto-parallel view.

If the camera roll is minimal in the video sequence and the altitude is high, then the fronto-parallel transformations are accurate enough to assume a pure translation model for the view-normalized image sequence. This simplified motion model due to the metadata enables mosaicking of long sequences with minimal projective distortion. To compare qualitatively the extent of the distortion in a mosaicked sequence with and without metadata, we tracked KLT features in an aerial sequence and its fronto-parallel equivalent. We used these feature tracks along with RANSAC to compute the interframe transformations using a homography model for the original sequence and a translation model for the metadata normalized sequence. Figure 4.4 illustrates the mosaicks obtained using the translation model and homography model. Using the constrained motion model imposed by the metadata, we are able to mosaic long sequences of around 1700 frames without distortion and limited buildup of error [21].

We use the fronto-parallel sequence for moving object detection using the joint tracking and segmentation algorithm. We present results on three different video sequences in the dataset.



(a)



(b)

Figure 4.4: This figure qualitatively compares the mosaics obtained by stabilizing the original sequence using a homography model and the fronto-parallel sequence using a translation model. Figure (a) shows the mosaic obtained using the translation model. It is one part of a long mosaic of around 550 frames. Figure (b) shows the mosaic obtained using the homography model. The distortion is clear towards the right side of the mosaic because of the gradual build up of errors in homography estimation.

4.2.2 Video sequence 1

The first video sequence consists of a camera moving over a runway, surveying vehicles moving on the runway. The sequence consists of moving vehicles entering and leaving the field-of-view. The background may be modeled as a plane since there is no parallax due to 3D objects on the air strip. Stabilizing the sequence is difficult due to the textureless background. The presence of repeated textures on the road makes the registration problem even harder (since the regions of significant gradients repeat themselves in the image).

We compared the performance of the proposed joint feature-tracking and segmentation-based registration approach with the intensity-based approach and individual feature tracking approach. We select 100 frames of the sequence and use the metadata to obtain fronto-parallel views. We detected the prominent lines on the images and used them to correct the small rotation-errors in camera-roll due to the metadata. This ensured that the fronto-parallel sequence obeyed a pure-translation motion model throughout.

For the intensity-based method, we used the inverse compositional registration algorithm studied in the earlier section to register consecutive sequences of frames. This algorithm consistently registered the moving objects between consecutive frames, since the gradients on the moving objects were much more prominent than those on the background.

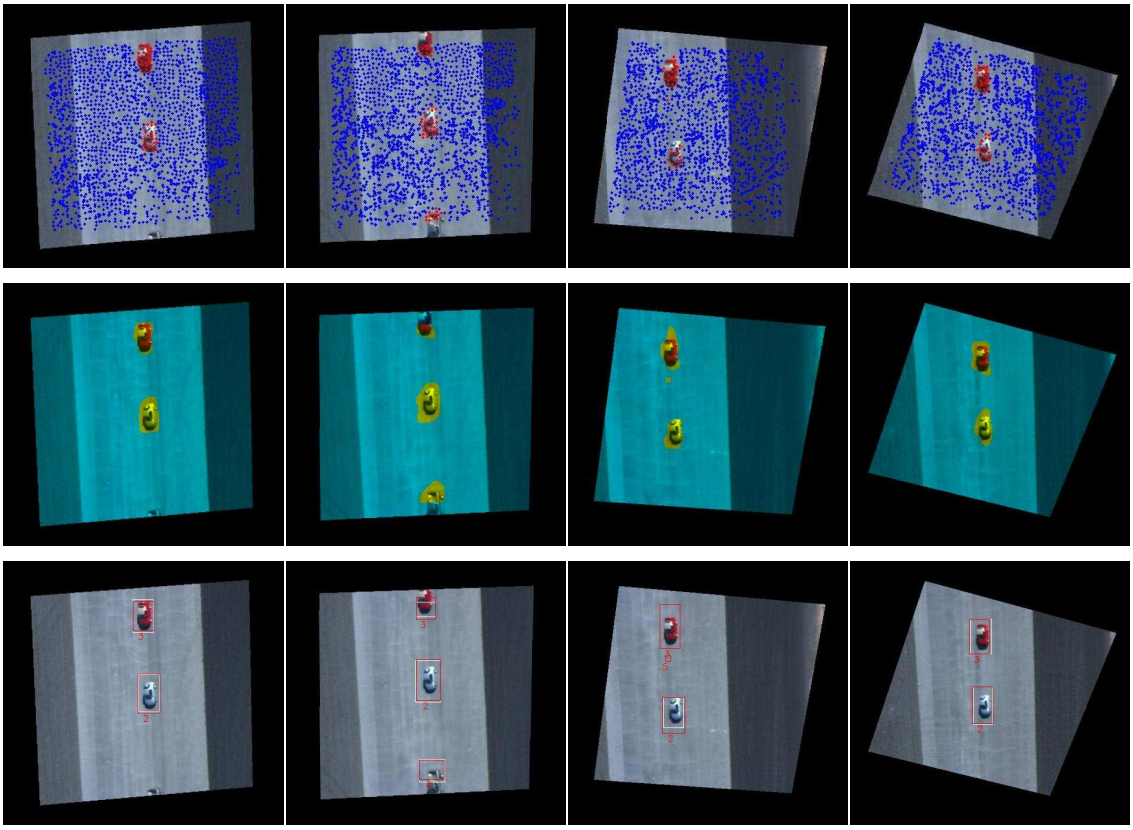
We tracked KLT features independently through the sequence and solved for translation using RANSAC. We found that the presence of repeated textures in the sequence was a serious problem for many feature points. In practice, we experienced registration

failures in multiple images of the sequence due to registration of the moving objects as opposed to the background.

Finally, we used the proposed joint tracking and segmentation algorithm to solve for inter-frame displacement parameters and motion segmentations. We initialized the segmentation by choosing 2500 features on the first frame automatically and labeling all the features as belonging to the background. It must be emphasized that there was no manual initialization or segmentation of any kind. This initialization of all features to background incorrectly labels the features on the three moving vehicles as background features. The joint tracking using robust cost function solves for the accurate displacement and enables the classification block to correctly identify the mislabeled features. The mislabeled features are subsequently removed and new features are initialized and tracked on the moving objects.

Figure 4.5 illustrates the results of our algorithm on the sequence. For four different frames, we show the set of classified features overlaid on the images, the dense segmentation and the tracking result. The classification of features into background and foreground is extremely good throughout the sequence and we found only one feature on the background that was misclassified to be on the foreground. We used the set of labeled features to infer the dense segmentation and obtain the motion blobs. We found that throughout the sequence, there was only one false moving target initialization in one frame (due to a misclassified feature) that was quickly eliminated in subsequent frames. Column (c) of figure 4.5 illustrates the frame with the false moving target initialization. In comparison, a background subtraction followed by a blob-based tracking algorithm [2] resulted in 70 moving target initializations. Figure 4.6 illustrates a mosaic of the sequence

with the moving objects detected by the algorithm removed from the frames. The mosaic has a very small artifact as a red patch belonging to a vehicle that was not segmented out completely in one frame.



(a) (b) (c) (d)

Figure 4.5: This figure shows results on four frames of the 100 frame long VIVID Video Sequence 1. Column (a) shows results on frame 14, column (b) shows results on frame 23, column (c) shows results on frame 50 and column (d) shows results on frame 92. The top row in each column overlays the feature points tracked on each frame. The blue points are classified to be on the background and the red points are classified to be on the moving objects. As the figures illustrate, the segmentation is very accurate and is much better than individual KLT tracking followed by RANSAC based background feature selection. The middle row in each column illustrates the dense segmentations inferred from the feature segmentations. The background and moving objects are plotted on different color channels for illustration. The bottom row in each column illustrates the tracked boxes on the images which is the result of a blob tracking algorithm [2]. Column (c) shows the only feature on the background which is misclassified to be on the foreground. There is a false moving target initialization due to this feature but this is quickly removed by the algorithm.

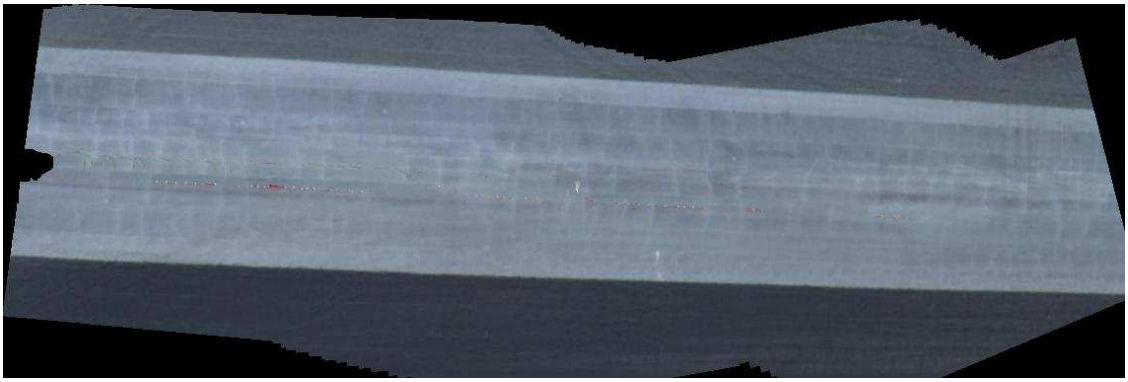


Figure 4.6: This figure shows the mosaic of 100 frames of the VIVID sequence 1. The moving objects are removed from the individual frames before mosaicking. The absence of moving object trails on the mosaic illustrates the accuracy of motion segmentation in this sequence.

4.2.3 Video sequence 2

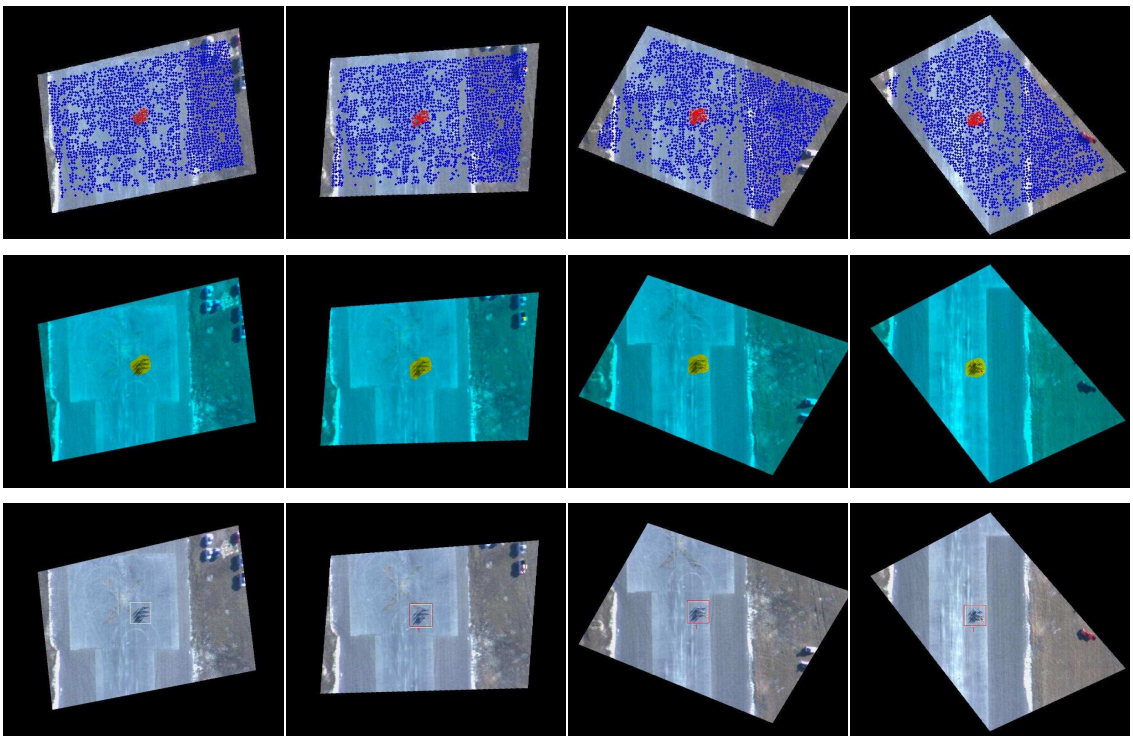
The second sequence consists of a group of humans walking together on a runway. This sequence had sufficient features on the images as well as gradient information for intensity-based stabilization. The group of people move as a single moving blob and always lie within the field of view in the sequence. The chosen sequence was 190 frames long. We compared the performance of intensity and feature based registrations with the proposed joint tracking and segmentation approach for the purpose of stabilization and moving object detection.

The intensity-based algorithm registered the background regions of all the frames as expected. We used background subtraction techniques [2] on the stabilized sequence to detect moving objects. On the entire sequence, 45 false positives were detected due to slight misregistrations near edges. In addition, the motion blobs corresponding to the true moving objects were broken into multiple connected components for most of the frames. Among the 190 frames, 49 of them had multiple connected components for the single

moving object, and 55 of them had no detected blobs corresponding to the true moving object. The absence of blobs was because the group of people moved slowly, and there were homogeneous regions of relatively unvarying intensity within the blob.

In the feature-based algorithm, we track KLT features individually, use RANSAC to identify a set of background tracks, and use them to solve for motion model parameters between consecutive frames. We stabilize the sequence using the estimated homographies and then identify moving blobs [2]. On the entire sequence, there were 21 frames where no blob was detected corresponding to the moving object. One single blob was detected in 88 of the frames, $2-3$ blobs in 65 frames and $4-5$ blobs in 16 frames. There were 34 false positive moving objects that were tracked through the sequence.

When the joint tracking and segmentation algorithm was applied to the sequence, it detected a single motion blob that enclosed the entire object in all the frames. There were only 5 false positive objects detected, 3 of which were features in some frames lying on off-planar objects. These features represented the parallax-induced motion between frames of 3D objects on the plane. Only 2 features clearly lying on the plane were misclassified by the algorithm to lie on moving objects. Figure 4.7 illustrates the results of our algorithm on the sequence. For four different frames, we show the set of classified features overlaid on the images, the dense segmentation and the tracking result. The classification of features into background and foreground is extremely good throughout the sequence. Column (b) of figure 4.7 illustrates a frame with a feature on a stationary vehicle that was classified as moving.



(a) (b) (c) (d)

Figure 4.7: This figure shows results on four frames of the 190 frame long VIVID Video Sequence 2. Column (a) shows results on frame 1, column (b) shows results on frame 24, column (c) shows results on frame 91 and column (d) shows results on frame 175. The top row in each column overlays the feature points tracked on each frame. The blue points are classified to be on the background and the red points are classified to be on the moving objects. As the figures illustrate, the segmentation is very accurate and is much better than the results from individual KLT tracking followed by RANSAC-based background feature selection. The middle row in each column illustrates the dense segmentations inferred from the feature segmentations. The background and moving objects are plotted on different color channels for illustration. The bottom row in each column illustrates the tracked boxes on the images which is the result of a blob tracking algorithm [2]. Column (b) shows a feature on a stationary vehicle which is misclassified to be a a moving feature due to parallax-induced motion.

Figure 4.8: This figure shows the mosaic of 190 frames of the VIVID sequence 2. The moving objects are removed from the individual frames before mosaicking. The absence of moving object trails on the mosaic illustrates the accuracy of motion segmentation in this sequence.

4.2.4 Video sequence 3

The third sequence captures a scenario where a set of people and vehicles are moving independently on a road. The sequence consists of 245 frames. The camera surveys this scene in such a way that the moving objects continuously move in and out of the field of view of the camera. The ground plane has very little gradient information in the Y direction which is a challenge for intensity-based stabilization algorithms and also for feature trackers. In addition, since the people move independently, each of them occupied a very small area within the image plane which was a challenge.

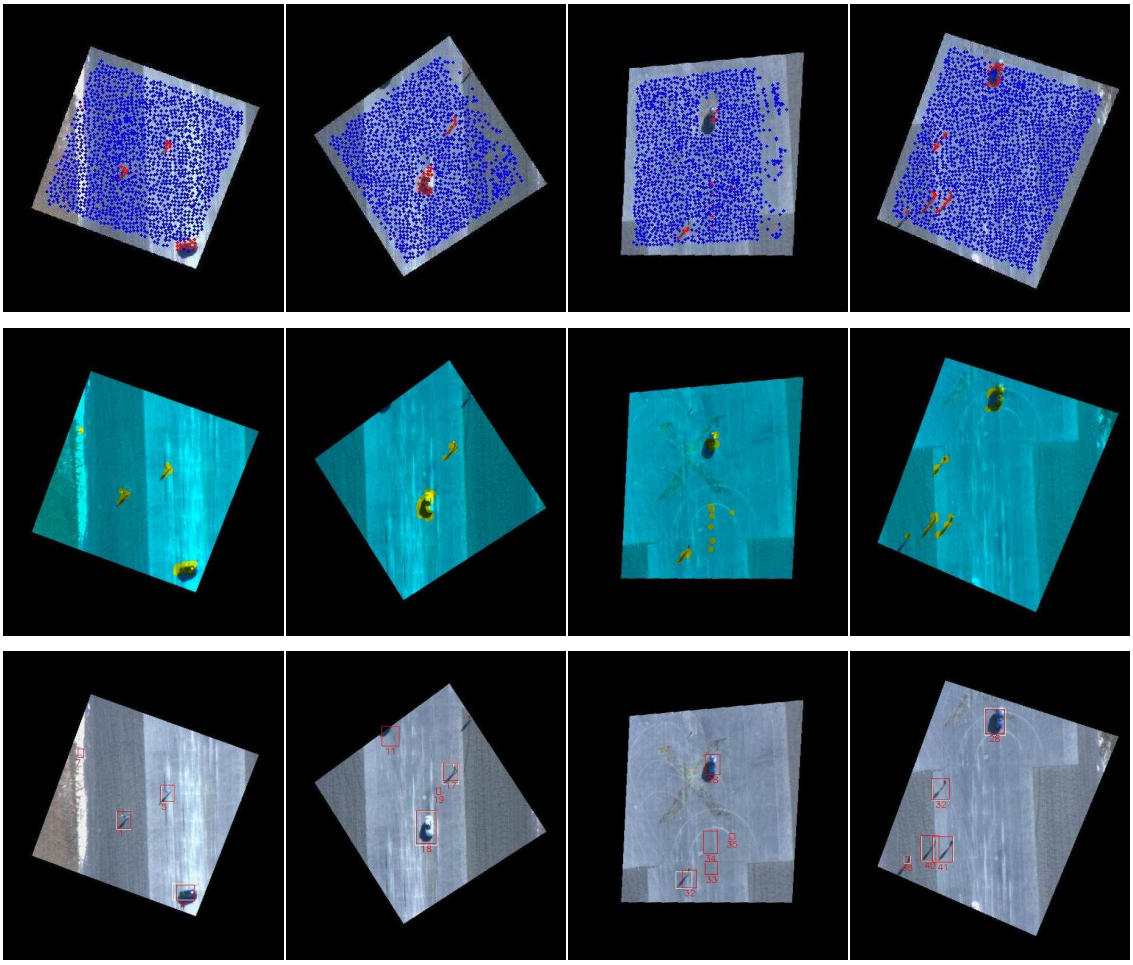
The intensity-based stabilization algorithm failed for most of the images since the image motion was in the Y direction and the image did not have predominant gradients along that direction. The feature-based stabilization algorithm performed much better although there were significant registration errors in some pairs of images. On the entire sequence, there were 106 cases of missing blobs corresponding to moving objects. In

many of the cases where a blob was detected corresponding to the moving people, the blob was very small and did not enclose the humans. There were 47 false target initializations throughout the sequence for blobs which showed up on the background due to misregistration and errors in background subtraction.

When the joint tracking and segmentation algorithm was applied, all the moving objects were accurately detected in all the frames. There were 18 false positives due to isolated background features that were classified as outliers. However, most of these features were quickly removed by the algorithm since they obeyed the background model in subsequent frames. Among the false positives, only 3 of them were from background features that were continually classified as moving. With this very simple filtering rule, the number of false positives of our algorithm got down to 3. These results illustrate the competence of the proposed algorithm in stabilizing and detecting small moving objects from aerial videos.

4.3 Conclusions

This chapter presents a joint tracking and segmentation algorithm to exploit motion coherency of the background as well as solve for the class labels of features. Although algorithms exist for tracking features jointly, the idea of incorporating segmentation within this framework and using the feature dissimilarities to infer membership probabilities yields robust results. The proposed approach produces high accuracy labeled feature tracks in a sequence that are uniformly distributed. This enables us to infer dense pixel-wise motion segmentation that is useful in moving object initialization. We demonstrate



(a) (b) (c) (d)

Figure 4.9: This figure shows results on four frames of the 245 frame long VIVID Video Sequence 3. Column (a) shows results on frame 70, column (b) shows results on frame 142, column (c) shows results on frame 196 and column (d) shows results on frame 235. The top row in each column overlays the feature points tracked on each frame. The blue points are classified to be on the background and the red points are classified to be on the moving objects. As the figures illustrate, the segmentation is very accurate and is much better than individual KLT tracking followed by RANSAC-based background feature selection. The middle row in each column illustrates the dense segmentations inferred from the feature segmentations. The background and moving objects are plotted on different color channels for illustration. The bottom row in each column illustrates the tracked boxes on the images which is the result of a blob tracking algorithm [2]. Column (c) shows a frame with a few features on the background which are misclassified to be on the foreground. There are a false moving target initializations due to these feature but they are quickly removed by the algorithm in the subsequent frame.

robust results in several challenging video sequences in the VIVID dataset. We also qualitatively compare the improvement in image mosaics obtained using the information provided by associated metadata containing telemetry information.

Chapter 5

Fast Bilinear Structure from Motion

In an urban environment, the additional sensors are not accurate enough to measure the location precisely. In the presence of tall buildings, it is difficult to triangulate the GPS position because of obstacles to the line-of-sight. In this chapter, we study the effect of additional information, in the form of measurements of a direction vector and the height of the camera center from a plane perpendicular to this vector. This type of side information is frequently available and accurately measurable in several real-world scenarios such as when onboard inertial measurements are available or when a dominant plane is present.

Inertial sensors like the inclinometer or gravitational sensors can provide sensing of a certain direction while altimeters frequently found on UAVs can provide the required height information. For example, when there is negligible camera acceleration, an accelerometer measures the gravity and we can filter out the IMU measurements to get good estimates of the gravity vector. When we do not have side information but we observe a dominant plane in the scene, we can use the homographies between multiple views to obtain estimates of the ground plane normal and height using a decomposition technique. In either case, we will show that this side information constrains the ill-posed SfM problem in such a manner that the SfM equations become similar to a bilinear form in its unknowns. We then describe a fast iterative procedure much like bilinear solvers that can robustly solve for the SfM unknowns by minimizing the reprojection error [69, 70].

Figure 5.1 illustrates the main computational steps of the algorithm.

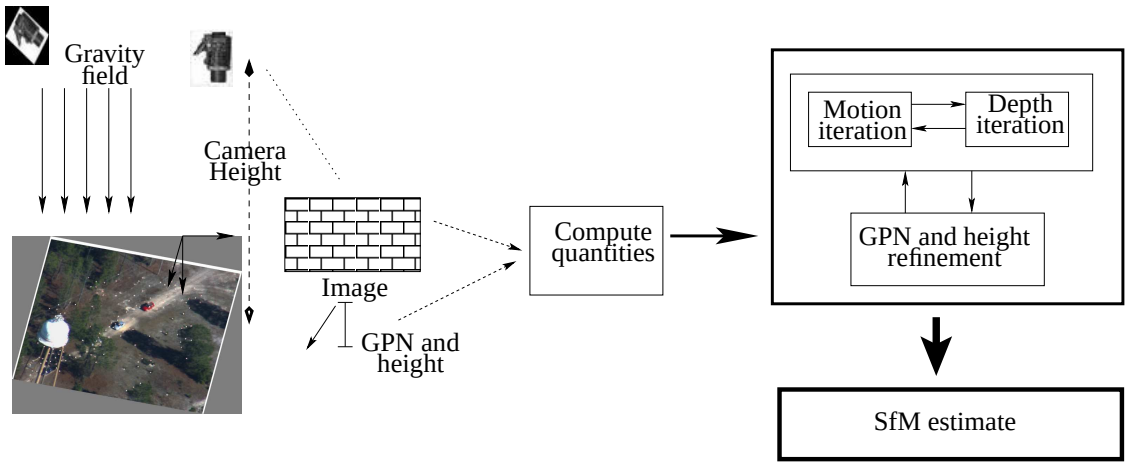


Figure 5.1: An illustration of the problem setting and the main computational steps. The image shows a typical scene with a ground plane and some static and moving objects on it. The gravity field is shown, which can be assumed perpendicular to the ground plane. A camera moves over the scene and gives an image sequence. We may also have measurements of the gravity vector from an IMU or sensing of the plane normal by additional means. We use these additional measurements to simplify the structure and motion computations.

We assume that we have measurements of the gravity vector and the height in the camera coordinate system along with every image in the sequence. This form of additional information is commonly available in the following scenarios:

1. In the problem of urban modeling using a ground-based platform, we typically have a car with a set of cameras mounted on top. These cameras record images of the city while the car is moving. The car also has other sensors such as an inertial measurement unit, GPS receivers and sensors that measure the speed from the wheel rotation rates. Figure 1.1(a) shows an image of such a car with attached sensors. Since the vehicle moves on the ground, its height does not vary much. In addition, since the rotation of the car is primarily along only the vertical axis, the gravity vector (vertical direction) does not vary much when measured in the camera

coordinate system. Both these quantities can be measured accurately using onboard sensors.

2. With advances in Micro Air Vehicle (MAV) technology, MAVs are increasingly being used for surveying and mapping environments. Figure 1.1(b) shows an illustration of an MAV surveying a scene. These MAVs typically have IMUs, altimeters etc that sense the required additional information. Our algorithm can be used in this situation to solve for the structure and motion from video sequences.

3. In the scenario where a UAV observes the ground plane from moderate to high altitudes, we describe how we can derive the necessary additional information from the homographies induced by the plane from multiple views. We can decompose these homographies [71] to solve for the ground-plane normal vector and heights which can be used as initial solutions in our algorithm.

There are other kinds of additional information that can be leveraged to develop video processing algorithms. For instance, the raw inertial measurements sense the acceleration and rotation rates that can potentially be fused with inertial sensors. Range data from laser scanners found in ground-based platforms can be used with images for highly accurate mapping. However, these additional data are outside the scope of the current work.

5.1 Problem Formulation

We choose a World Coordinate System (WCS) with the Z axis along the vertical direction, and the X and Y axes perpendicular to this vector. If a ground plane is present

in the scene, the Z axis becomes the normal vector to the plane, and the X and Y axes are on the plane. The Camera Coordinate System (CCS) is chosen with the Z axis along the optical camera axis and the X and Y axes along the usual image axes. The transformation between these two coordinate systems at any instant can be written as $P_w = R_{c2w}P_c + T_{c2w}$. Here, P is a point whose coordinates are represented in the WCS by P_w, and in the CCS by P_c. Figure 5.2 illustrates the various coordinate systems as described above.

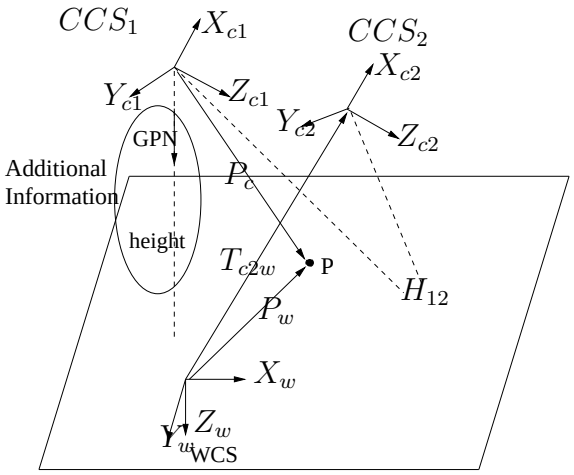


Figure 5.2: The figure shows an illustration of the coordinate systems in the problem setting. The world coordinate system and the camera coordinate systems corresponding to two viewpoints is shown in the figure. GPN illustrates the ground plane normal vector which coincides with the vertical or direction of gravity in most problem scenarios. H_{12} denotes the homography induced by the plane between the two views. P denotes a point in the world whose coordinates in the WCS and CCS are P_w and P_c respectively, with $P_w = R_{c2w}P_c + T_{c2w}$. As the camera moves, we obtain images of the world. We also assume that we have measurements of the GPN and the camera height with every image, as additional information.

We assume that we have measurements of a certain direction in CCS corresponding to every image in the sequence. This direction could be the gravity vector which can be sensed using inclinometers, or the normal vector to a ground plane which can be obtained using the homographies. We also assume that we have measurements or estimates of the heights of the camera along the direction of the reference vector from a plane perpendic-

ular to the vector.

A known reference vector in an unknown camera coordinate camera system fixes two degrees of freedom of the rotation matrix R_{c2w}. The unknown component is the rotation of the CCS about an axis parallel to the reference direction vector. The full rotation matrix can be shown to be split uniquely as $R_{c2w} = R_p R_g$, where R_p is the rotation along the direction vector, and R_g is along an axis perpendicular to this vector. We are now concerned with the estimation of the rotation along the direction vector (R_p), and the translations along a plane perpendicular to this vector (x and y components of T_{c2w}) in addition to the 3D locations of the world points. In the following, we refer to in-plane motion as the component of translation parallel to the $X - Y$ world plane and rotation about the Z-axis (R_p). The out-of-plane motion is the Z-axis translation (z component of T_{c2w}), and the rotation R_g that changes the reference vector orientation in the local coordinate system (CCS). Table 5.1 lists all the notations used in the following derivations.

We write the transformation between the WCS and the CCS as:

$$P_w^i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = R_p^{(t)} R_g^{(t)} \lambda_{ti} \begin{bmatrix} x_{ti} \\ y_{ti} \\ 1 \end{bmatrix} + T_{c2w}^{(t)} \tag{5.1}$$

where the camera-to-world rotation matrix has been factorized into its two components as $R_{c2w}^{(t)} = R_p^{(t)} R_g^{(t)}$. Here $T_{c2w}^{(t)} = [T_x^{(t)}, T_y^{(t)}, T_z^{(t)}]$ where $T_z^{(t)}$ is the height of the camera. $[x_i, y_i, 1]^T$ is the image feature in homogeneous coordinates, which has been normalized for the calibration matrix. P_w^i denotes the coordinates of the i^{th} point in the

Notation	Meaning
P	Coordinates of a point
subscript w	Quantity expressed in WCS
subscript c	Quantity expressed in CCS
subscript c2w	Camera-to-world transformation
subscript ti	Coordinates at frame t and feature i
index t	Frame number
index i	Feature number
$T^t_{\{x,y,z\}}$	Translation coordinates at frame t
R^t_g	Out-of-plane rotation at frame t
R^t_p	In-plane rotation at frame t
m	Number of frames
n	Number of points

Table 5.1: This table lists common notations used in the derivations for purposes of readability.

WCS. From the additional measurements, we have estimates (initial values) of $R_g^{(t)}$ and $T_z^{(t)}$. Using this information, we can rewrite (5.1) as

$$P_w^i = \begin{pmatrix} \cos\theta_t & \sin\theta_t & 0 \\ -\sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{pmatrix} \lambda_{ti} \begin{bmatrix} u_{ti} \\ v_{ti} \\ w_{ti} \end{bmatrix} + T_{c2w}^{(t)} \tag{5.2}$$

where $[u_{ti}, v_{ti}, w_{ti}]^T = R_g^{(t)}[x_i, y_i, 1]^T$, and $R_g^{(t)}$ is computed from the reference vector in the side information. $R_g^{(t)}$ is the rotation matrix that rotates the reference vector expressed in the CCS to the reference vector expressed in the WCS.

We rearrange (5.2) to get (5.3) that relates the coordinates of the feature point in frame t and feature i to the world coordinates and the camera positions.

$$\lambda_{ti} \begin{bmatrix} u_{ti} \\ v_{ti} \\ w_{ti} \end{bmatrix} = \begin{pmatrix} \cos\theta_t & -\sin\theta_t & 0 \\ \sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} X_i - T_x^{(t)} \\ Y_i - T_y^{(t)} \\ Z_i - T_z^{(t)} \end{bmatrix} \tag{5.3}$$

We eliminate the projective depth λ_{ti} by taking ratios of the quantities as shown in (5.4)

$$\begin{bmatrix} u_{ti}/w_{ti} \\ v_{ti}/w_{ti} \end{bmatrix} = \begin{pmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{pmatrix} \begin{bmatrix} \frac{X_i - T_x^{(t)}}{Z_i - T_z^{(t)}} \\ \frac{Y_i - T_y^{(t)}}{Z_i - T_z^{(t)}} \end{bmatrix} \tag{5.4}$$

We rearrange (5.4) by multiplying both sides by $(Z_i - T_z)$ to obtain (5.5)

$$\begin{bmatrix} u_{ti}/w_{ti} \\ v_{ti}/w_{ti} \end{bmatrix} \cdot (Z_i - T_z^{(t)}) = \begin{pmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{pmatrix} \begin{bmatrix} X_i - T_x^{(t)} \\ Y_i - T_y^{(t)} \end{bmatrix} \tag{5.5}$$

Let us now assume that we have n feature points that are observed in m frames. We have equations similar to (5.5) for each feature point in every frame where it is visible. We accumulate (5.5) for all the feature points in all views and write it in the factorization

format as shown in (5.6).

$$\begin{bmatrix} \frac{u_{11}}{w_{11}} & \frac{u_{12}}{w_{12}} & \dots & \frac{u_{1n}}{w_{1n}} \\ \frac{v_{11}}{w_{11}} & \frac{v_{12}}{w_{12}} & \dots & \frac{v_{1n}}{w_{1n}} \\ \vdots & & \ddots & \dots \\ \frac{u_{m1}}{w_{m1}} & \frac{u_{m2}}{w_{m2}} & \dots & \frac{u_{mn}}{w_{mn}} \\ \frac{v_{m1}}{w_{m1}} & \frac{v_{m2}}{w_{m2}} & \dots & \frac{v_{mn}}{w_{mn}} \end{bmatrix} \begin{bmatrix} Z_1 & 0 & \dots & 0 \\ 0 & Z_2 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & Z_n \end{bmatrix} - \begin{bmatrix} T_z^{(1)} & 0 & \dots & 0 & 0 \\ 0 & T_z^{(1)} & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & T_z^{(m)} & 0 \\ 0 & 0 & \dots & 0 & T_z^{(m)} \end{bmatrix} \begin{bmatrix} \frac{u_{11}}{w_{11}} & \dots & \frac{u_{1n}}{w_{1n}} \\ \frac{v_{11}}{w_{11}} & \dots & \frac{v_{1n}}{w_{1n}} \\ \vdots & \ddots & \dots \\ \frac{u_{m1}}{w_{m1}} & \dots & \frac{u_{mn}}{w_{mn}} \\ \frac{v_{m1}}{w_{m1}} & \dots & \frac{v_{mn}}{w_{mn}} \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & t_x^{(1)} \\ \sin\theta_1 & \cos\theta_1 & t_y^{(1)} \\ \dots & & \\ \cos\theta_m & -\sin\theta_m & t_x^{(m)} \\ \sin\theta_m & \cos\theta_m & t_y^{(m)} \end{bmatrix} \begin{bmatrix} X_1 & \dots & X_n \\ Y_1 & \dots & Y_n \\ 1 & \dots & 1 \end{bmatrix} \tag{5.6}$$

We denote the measurement matrix as A, the diagonal matrix of camera heights as $\bar{T}_z$, and the product $\bar{T}_z A = B$. We denote the diagonal matrix of heights of feature points from the ground plane as Z. The motion matrix on the right hand side is M and the shape matrix is S. We can rewrite (5.6) concisely as: $AZ - B = MS$. Each column of this matrix equation specifies the relation between the projections of a single point in all the views. Each pair of rows specifies the relation between the projections of all the points in a single view. In (5.4), the quantities $T_x^{(t)}$ and $T_y^{(t)}$ refer to the x and y components of translation measured along the WCS axes. In (5.6), the variables $t_x^{(t)} = -cos\theta_t T_x^{(t)} + sin\theta_t T_y^{(t)}$ and $t_y^{(t)} = -sin\theta_t T_x^{(t)} - cos\theta_t T_y^{(t)}$ refer to the same quantities measured in the CCS axes. This change of variables is done to enable a factorization into the motion and shape matrices as shown. Note that our unknowns are the matrices $\{M, S, Z\}$ and (5.6) looks similar to a bilinear system in the elements of these matrices.

In practice, we have measurement errors in image features or in additional measurements or both, and the measurement matrix A is not known exactly. We solve for the unknown parameters (M, S, Z) by optimizing an objective function which is the Frobenius norm of the difference between the matrices on each side of (5.6). The cost function is written as follows:

$$E = \|A \cdot Z - \bar{T}_z A - M \cdot S\|^2 \tag{5.7}$$

In the above, because features may not be observed in all frames, some entries of the measurement matrix are unknown (not measurable). However, in solving for the unknowns using our algorithm, we do not need all entries of the measurement matrix. In this respect, it has an advantage over factorization-based approaches.

This formulation yields an optimal solution in the maximum-likelihood sense when Gaussian noise is added to the right hand side of (5.6). In practice, we can make a reasonable assumption that the image feature locations are corrupted by Gaussian noise. However, the geometric transformations and the non-linearity in the rotation matrices do not preserve the Gaussian nature of the error, and it is difficult to characterize the distribution of the error in measurements in (5.6).

The scale ambiguity in SfM implies that we can decrease the error (5.7) by scaling the camera positions and $3D$ coordinates of points by any number larger than one. Since we refine all components of motion in our algorithm, to guard against trivial reductions in error functions because of such scaling, we impose the constraint: $\left\|\bar{T}_z\right\| = k$ where k is a constant.

5.2 Fast Bilinear Estimation of SfM

We observe that (5.6) is similar to a bilinear form in structure and in-plane motion. We solve for the unknowns by minimizing the cost function $E = \|A \cdot Z - B - M \cdot S\|^2$ where $\|\cdot\|^2$ denotes the Frobenius norm.

Our approach is to solve for the unknowns in an iterative manner, alternating iterations where (1) the motion parameters are kept fixed and structure parameters are estimated, with iterations where (2) the structure is fixed and motion parameters are estimated. This approach belongs to a class of SfM techniques known as resection-intersection methods [56, 72]

5.2.1 Structure Iterations

We can rewrite the cost function (5.7) as a sum of terms corresponding to each feature point j as follows: $E = \sum_{j=1}^{n} E_j^d$, where E_j^d corresponds to the contribution to the cost function from the j^{th} feature point, and the superscript d is used to note that the total error is split into terms corresponding to the depth of each feature point.

We pick the j^{th} column of the matrix equation (5.6) and write the corresponding contribution to the error as follows.

$$E_j^d = \left\| \begin{bmatrix} u_{1j}(Z_j - T_z^1)/w_{1j} \\ v_{1j}(Z_j - T_z^1)/w_{1j} \\ u_{2j}(Z_j - T_z^2)/w_{2j} \\ \vdots \\ v_{mj}(Z_j - T_z^m)/w_{mj} \end{bmatrix} - \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & t_x^1 \\ \sin\theta_1 & \cos\theta_1 & t_y^1 \\ \cdots & & \\ \cos\theta_m & -\sin\theta_m & t_x^m \\ \sin\theta_m & \cos\theta_m & t_y^m \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ 1 \end{bmatrix} \right\|^2 \tag{5.8}$$

We rewrite (5.8) to obtain

$$E_j^d = \left\| \begin{bmatrix} M(:,1) & M(:,2) & -A(:,i) \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} + (B(:,j) + M(:,3)) \right\|^2 \tag{5.9}$$

In structure iterations, we minimize the error contribution E_j^d corresponding to each 3D point indexed by $j \in (1, \cdots, n)$. In (5.9), $M(:,1)$ and $M(:,2)$ are the first and second columns of the motion matrix, containing the cosine and sine terms. $M(:,3)$ contains the in-plane components of translation. $A(:,j)$ and $B(:,j)$ are the j^{th} columns of the matrices A and B respectively. Recall that B is the product of the heights matrix T_z, and the measurement matrix A. We minimize (5.9) w.r.t (X_j, Y_j, Z_j). The cost function is a linear system in the variables (X_j, Y_j, Z_j) and the minimum is obtained by solving the following linear system using linear least squares.

$$\begin{bmatrix} M(:,1) & M(:,2) & -A(:,j) \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} = -B(:,j) - M(:,3) \tag{5.10}$$

5.2.2 Motion Iterations

As in the case of depth refinement iterations, we can rewrite the cost function (5.7) as a sum of terms corresponding to each frame as follows: $E = \sum_{i=1}^{m} E_i^m$, where E_i^m corresponds to the contribution to the total cost function due to the motion variables of the i^{th} frame, and the superscript m is used to note that the total error is split into terms corresponding to the motion parameters of each frame. We extract the $(2i-1)^{th}$ and $(2i)^{th}$ rows from the matrix equation (5.6) and write the corresponding contribution to the

error as

$$E_i^m = \left\| C(2i-1,:) - \begin{bmatrix} \cos\theta_i & -\sin\theta_i & t_x^i \end{bmatrix} \cdot S \right\|^2 + \left\| C(2i,:) - \begin{bmatrix} \sin\theta_i & \cos\theta_i & t_y^i \end{bmatrix} \cdot S \right\|^2 \quad (5.11)$$

where, the $C = AZ - B$ denotes the left hand side of (5.6). We need to solve for $\{\theta_i, t_x^i, t_y^i\}$ by minimizing (5.11). We set the derivative of E_i^m w.r.t (t_x^i, t_y^i) to zero to obtain

$$t_x^i = \frac{1}{n}\sum_{k=1}^{n} C(2i-1,k) - \frac{1}{n}\sum_{k=1}^{n} X_k cos\theta_i + \frac{1}{n}\sum_{k=1}^{n} Y_k sin\theta_i \quad (5.12)$$

$$t_y^i = \frac{1}{n}\sum_{k=1}^{n} C(2i,k) - \frac{1}{n}\sum_{k=1}^{n} X_k sin\theta_i - \frac{1}{n}\sum_{k=1}^{n} Y_k cos\theta_i \quad (5.13)$$

Denoting μ_{2i-1} and μ_{2i} as the means of $C(2i-1,:)$ and $C(2i,:)$ respectively, and μ_X and μ_Y as the means of the X and Y coordinates of feature points, we can write the solution of t_x^i and t_y^i as

$$t_x^i = \mu_{2i-1} - \mu_X \cos\theta_i + \mu_Y \sin\theta_i \quad (5.14)$$

$$t_y^i = \mu_{2i} - \mu_Y \cos\theta_i - \mu_X \sin\theta_i \quad (5.15)$$

We substitute the solution (5.15) in (5.11) and obtain

$$E_i^m = \left\| \left(\vec{X} - \mu_X\right)\cos\theta_i - \left(\vec{Y} - \mu_Y\right)\sin\theta_i - \left(C(2i-1,:)^T - \mu_{2i-1}\right) \right\|^2 + \left\| \left(\vec{X} - \mu_X\right)\sin\theta_i + \left(\vec{Y} - \mu_Y\right)\cos\theta_i - \left(C(2i,:)^T - \mu_{2i}\right) \right\|^2 \quad (5.16)$$

where $\vec{X}$ and $\vec{Y}$ denote column vectors containing the X and Y coordinates of all the points. We set the derivative w.r.t θ_i to zero and simplify to obtain

$$\sum_{k=1}^{n} \left(C(2i-1,k) - \mu_{2i-1}\right)\left(\left(X_k - \mu_X\right)\sin\theta_i + \left(Y_k - \mu_Y\right)\cos\theta_i\right) + \sum_{k=1}^{n} \left(C(2i,k) - \mu_{2i}\right)\left(-\left(X_k - \mu_X\right)\cos\theta_i + \left(Y_k - \mu_Y\right)\sin\theta_i\right) = 0 \quad (5.17)$$

We can simplify (5.17) to obtain

$$\tan\theta_i = \frac{(C(2i,:) - \mu_{2i}) \cdot \left(\vec{X} - \mu_X\right) - (C(2i-1,:) - \mu_{2i-1}) \cdot \left(\vec{Y} - \mu_Y\right)}{(C(2i-1,:) - \mu_{2i-1}) \cdot \left(\vec{X} - \mu_X\right) + (C(2i,:) - \mu_{2i}) \cdot \left(\vec{Y} - \mu_Y\right)} \tag{5.18}$$

We obtain two possible solutions for θ_i from (5.18). One of them is a point of local maxima and the other is a point of local minima. We choose the solution that corresponds to the local minima.

We have described a technique to iteratively estimate the 3D locations, and the in-plane components of motion. Each of the individual steps only involves solving a linear system. In practice, we iterate these two steps a certain number of times, or until a termination criterion is satisfied.

In both the depth and motion iterations, we solve linear systems that are of much smaller size compared to the size of systems that we need to solve in techniques like bundle adjustment etc. More details on the computational requirements are given in section 5.3.1

5.2.3 Out-of-plane motion refinement iterations

In case the measurements of the direction vector and the camera height are not accurate, we may refine these parameters to obtain a better SfM estimate. We describe a technique to refine these quantities. The idea is to refine the out-of-plane motion components so that the error (5.7) decreases.

From (5.5), we have

$$\begin{bmatrix} u_{ti}/w_{ti} \\ v_{ti}/w_{ti} \end{bmatrix} \cdot (Z_i - T_z^{(t)}) = \begin{pmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{pmatrix} \begin{bmatrix} X_i - T_x^{(t)} \\ Y_i - T_y^{(t)} \end{bmatrix} \tag{5.19}$$

Recall that (u_{ti}, v_{ti}, w_{ti}) is a function of the out-of-plane rotation $R_g^{(t)}$ based on the rela-

tion:

$$\begin{bmatrix} u_{ti} \\ v_{ti} \\ w_{ti} \end{bmatrix} = R_g^{(t)} \begin{bmatrix} x_{ti} \\ y_{ti} \\ 1 \end{bmatrix} \tag{5.20}$$

In this step, we hold the 3D locations P_w^i and the in-plane components of motion $\{\theta_t, T_x^t, T_y^t\}$ fixed to their earlier estimates from the previous step. We perform a nonlinear refinement of the height T_z^t, and the normal vector at each frame separately. Note that this implies that at any stage, we only refine three parameters simultaneously (in comparison to all the cameras and points for a full bundle adjustment).

We refine the parameters by minimizing the following error function obtained from (5.5)

$$E_{side} = \sum_{t=1}^{m} \left\| \begin{bmatrix} u_{ti}/w_{ti} \\ v_{ti}/w_{ti} \end{bmatrix} \cdot (Z_i - T_z^{(t)}) - \begin{pmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{pmatrix} \begin{bmatrix} X_i - T_x^{(t)} \\ Y_i - T_y^{(t)} \end{bmatrix} \right\|^2 \tag{5.21}$$

We minimize (5.21) using the Levenberg-Marquardt algorithm [39] to solve for the out-of-plane motion parameters. This minimization involves only three parameters at any stage (two for the out-of-plane motion and one for the camera height).

5.2.4 Motion Estimation for Moving Objects

We can incorporate the estimation of structure and motion of objects moving on a plane, in the above framework. The objects of interest can be translating and rotating on a plane, and we assume that we can measure the ground plane normal and height with respect to the same plane. We reformulate the problem into one where the object is stationary, and a separate camera is moving over each object. This approach is similar to some earlier works [73]. The world coordinate of a point on a moving object changes

depending on the motion of the object, and is therefore parameterized by time along with the feature number.

$$\begin{bmatrix} X_{ti} \\ Y_{ti} \\ Z_{ti} \end{bmatrix} = \begin{pmatrix} \cos\phi_t & \sin\phi_t & 0 \\ -\sin\phi_t & \cos\phi_t & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + \begin{bmatrix} D_x^t \\ D_y^t \\ 0 \end{bmatrix} \tag{5.22}$$

In (5.22), ϕ_t corresponds to the rotation of the object in frame t with respect to its initial orientation. (D_x^t, D_y^t) denote the displacement of the object in the WCS. The relation between image feature points and moving scene points is

$$\begin{bmatrix} X_{ti} \\ Y_{ti} \\ Z_{ti} \end{bmatrix} = \begin{pmatrix} \cos\theta_t & \sin\theta_t & 0 \\ -\sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{pmatrix} \lambda_{ti} \begin{bmatrix} u_{ti} \\ v_{ti} \\ w_{ti} \end{bmatrix} + \begin{bmatrix} T_x^{(t)} \\ T_y^{(t)} \\ T_z^{(t)} \end{bmatrix} \tag{5.23}$$

We can rewrite (5.23) in the same form as for static points as follows

$$\lambda_{ti} \begin{bmatrix} u_{ti} \\ v_{ti} \\ w_{ti} \end{bmatrix} = \begin{pmatrix} \cos\psi_t & -\sin\psi_t & 0 \\ \sin\psi_t & \cos\psi_t & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} X_i - C_x^t \\ Y_i - C_y^t \\ Z_i - T_z^{(t)} \end{bmatrix}$$

$$\begin{bmatrix} C_x^t \\ C_y^t \end{bmatrix} = \begin{bmatrix} \cos\phi_t & -\sin\phi_t \\ \sin\phi_t & \cos\phi_t \end{bmatrix} \begin{bmatrix} T_x^t - D_x^t \\ T_y^t - D_y^t \end{bmatrix}$$

$$\psi_t = \theta_t - \phi_t \tag{5.24}$$

Equations (5.24) say that the motion of the object can be transferred to the camera, and the object can be considered to be stationary. We can solve for (5.24) in the same framework as for static scene points, and the virtual camera motions can be estimated. The object motion (ϕ_t, D_x^t, D_y^t) can be solved from the real and virtual camera motions $(\theta_t, T_x^t, T_y^t, \psi_t, C_t^t, C_y^t)$ using (5.24).

We initialize the iterations for a moving object by assuming a fixed height for all feature points. In practice, we found that the technique was not very sensitive to this initialization in our sequences. In later sections we refer to the proposed method as FBSfM which is an acronym for Fast Bilinear Structure from Motion.

5.3 Analysis of FBSFM

5.3.1 Computational Complexity and Memory Requirements

The main computations in our algorithm are in the solution of the linear systems in the depth and motion iterations, and the refinement steps in the direction vector and height refinements. Suppose there are m views of n points, and that all points are visible in all views. To emphasize, this assumption is not necessary for the purposes of the algorithm but simplifies the process of quantifying the memory requirements.

As mentioned in [52], let $SVD(a, b) = 4ab^2 + 8b^3$ be the cost of carrying out a singular value decomposition of a matrix with a rows and b columns. The following are the main computational requirements:

- *Depth Iterations:* For each point, this involves solving a linear system of size $2m \times 4$ which is equivalent to a total cost of $n \times SVD(2m, 4)$, where n is the number of points.

- *Motion Iterations:* For each view, this involves performing the computations in (5.18, 5.15) which is equivalent to $4n$ multiplications and $6n$ additions and is therefore $O(n)$ in computational cost. This accumulates to a total cost of $m \times O(n)$,

where m is the number of views.

- *Direction vector and height refinement:* We need to update only 4 parameters and hence this requires the solution of a 4×4 linear system. For all the frames, this comes at a cost of $m \times SVD(4,4) = 768m$ per iteration.

Totally, each iteration of FBSfM has a computational cost of $O(mn)$. In comparison, one full bundle adjustment takes up computations of the order of $O(nm+nm^2+m^3)$. Rother's system involves solving a linear system of size $(3n+3m) \times (3nm)$ whose computational cost is $SVD(3n+3m, 3nm) = 108(m^3n^2 + m^2n^3) + 216m^3n^3$. We observe that the peak computation for our individual steps increases only of the order of $O(mn)$.

5.3.2 Discussion

The FBSfM algorithm derived in the last section has the flavor of iterative methods for projective structure from motion [56, 44]. Suppose we have n fixed points $P_1, P_2, \cdots, P_n$ observed by m cameras, we can write the projection of the j^{th} point on the i^{th} camera as:

$$p_{ij} = \frac{1}{z_{ij}} M_i P_j \tag{5.25}$$

where M_i denotes the 3×4 projective matrix associated with the i^{th} camera, and z_{ij} denotes the projective depth associated with the j^{th} point in the i^{th} camera.

In a typical BA algorithm, we usually minimize the reprojection error which is $E = \sum_{i,j} |p_{ij} - \frac{1}{z_{ij}} M_i P_j|^2$ and solve for the camera locations M_i and point locations P_j. This technique is commonly known as bundle adjustment and employs a non-linear minimization method to optimize the objective function (reprojection error). This objective function is highly non-linear and is difficult to minimize and also is reported to have

many local minima [72]. To simplify the minimization, many authors consider the simplified (but related) objective function: $E_{lin} = \sum_{i,j} |z_{ij}p_{ij} - M_i P_j|^2$. Many projective SfM algorithms in the literature minimize this objective function to solve for the SfM parameters.

The proposed algorithm is similarly derived by scaling (5.4) by the $(Z_i - T_z^{(t)})$. The difference is that unlike earlier techniques, we use the additional information to leverage the bilinear form in the Euclidean frame, without using slack variables. Simulation results suggest clear advantages in speed and accuracy of the proposed algorithm when the additional information has low error.

Convergence in error value: Our algorithm has three main iterations: depth iterations, motion iterations and side-information refinement iterations. Let $E^{(0)}$ be the value of the initial error. In the depth iterations, we find solutions for the (X, Y, Z) co-ordinates that minimize the error with all other variables fixed. If $E^{(1)}$ is the value of the error after the depth iterations, we have $E^{(1)} \leq E^{(0)}$. Next, in motion iterations, we find solutions for the camera in-plane motion parameters that minimize the error with all other variables fixed. If $E^{(2)}$ is the error after motion iterations, we have $E^{(2)} \leq E^{(1)}$. Similarly, if $E^{(3)}$ is the error after side-information refinement, we have $E^{(3)} \leq E^{(2)}$. Each iteration finds solutions for the variables that decreases the error, hence the error is non-increasing. Since the error is lower-bounded by zero, we conclude that the successive error values after each iteration converge to a value E^*.

Convergence of parameters: As noted in [56], the above argument does not imply that the parameters converge or that the error converges to a local minimum. We prove that the error converges to a local minimum, and the parameters converge to the

optimal solution. We make use of the Global Convergence Theorem from optimization theory [56]. We elaborate the proof in the appendix A of the dissertation.

Issue of trivial minima: Oliensis and Hartley [57] pointed out the issue of convergence to solutions that were "trivial" in the sense of generating nonsensical structure and motion solutions. In our algorithm, since we compute Euclidean structure and motion, we are not affected by the same problems. In particular, we do not need to deal with "projective depths" that are considered as independent variables in the optimization. In all of our experiments on synthetic and real sequences, we have observed that the iterations always converge to meaningful results.

Convergence rate and speed of convergence: Iterations of low computational complexity do not imply faster convergence rate or lower time-to-convergence. The proposed algorithm falls within the class of alternation techniques which are susceptible to flatlining and are slower than second-order newton methods in the average case [74]. We claim based on strong experimental evidence that the proposed approach surprisingly violates this conventional wisdom, for the *specific* case of large-sized problems starting from a *specific* class of initial solutions with low out-of-plane motion error and high levels of in-plane motion error.

5.4 Simulations

5.4.1 Implementation

For BA, we use the SBA solver code [40] implemented in C. This uses LM minimization for non-linear least squares. The normal equations are solved by using the Schur

complement to factor out the structure parameters and the resulting system is solved using LU decomposition for the update to the camera parameters. The proposed algorithm was also implemented in C with a MATLAB interface. The motion-structure alternation iterations and out-of-plane motion refinement iterations were written in C and the interface to switch between the two sets of iterations was written in MATLAB with system calls to the corresponding C executables. Routines from OpenCV were used to solve the linear systems corresponding to motion iterations. The LM implementation in C [75] was used for the non-linear least squares minimization corresponding to the out-of-plane motion refinement iterations. The Conjugate gradient (CG) implementation in C [76] was used in the experiments to compare with alternation for minimizing the bilinear system (5.7). Computation time was measured using our implementation of a nanosecond timer. The reported times include the total time taken to execute all the operations within each C implementation except disk I/O operations.

5.4.2 Reconstruction problem generation

A reconstruction problem was synthesized by generating N points uniformly distributed within the cube specified by $-20 \leq X \leq 20$, $-20 \leq Y \leq 20$ and $10 \leq Z \leq 40$. The coordinates of the camera locations were uniformly distributed in the cube specified by $-25 \leq X \leq 25$, $-25 \leq Y \leq 25$, $55 \leq Z \leq 105$. The choice of dimensions is for illustration, and in practice the dimensions were scaled according to convenience. The orientations of the cameras were chosen by first generating the directions of the principal camera axes and then choosing the rotation angle of the camera around this axis. The

principal camera axis was chosen by generating a random point in the $X - Y$ plane that specifies the point of intersection of the principal axis with this plane. The rotation angle of the camera axis around the principal axis was randomly chosen between 0 to 2π. This scheme for generating points and cameras ensured a wide variation in the camera matrices and point locations, to mitigate any potential bias in the reported results to the choice of reconstruction problem. Image feature points were obtained by reprojecting the 3D points on the cameras based on perspective projection. The parameters of the camera were: focal length $= 320$, principal point $= (320, 240)$ and image size $= 640 \times 480$. Moderate Gaussian noise was added to the reprojected image feature locations to simulate feature detection errors.

5.4.3 SfM Initialization

Initial motion estimates are obtained by perturbing the ground truth motion. The in-plane and out-of-plane motion components are perturbed separately, with high errors in in-plane components and low errors in out-of-plane motion components. The x and y locations are displaced by a vector that is oriented in a random direction in the XY plane, and whose length is a specified fraction of the maximum dimension of the reconstruction. For example, a 70% error in the in-plane translation means that the camera center was displaced by a vector of length 0.7×50 in the XY plane. A 60° error in the in-plane rotation angle θ means that θ was perturbed in either the clockwise or counter-clockwise direction by 60°. A 10% error in the out-of-plane translation means that each camera center was moved either in the positive or negative Z direction by 0.1×40. The

ground plane normal is perturbed by adding a random vector to it such that the new vector makes a pre-specified angle with the original vector. The initial values for the 3D coordinates of feature points are obtained by triangulating each point using the image feature locations and the initial camera motion parameters, after solving the linear system of a structure iteration in (5.10)). All algorithms are initialized with the same initial solution and executed on the identical machines under identical loading conditions when reporting comparative results.

5.4.4 Comparative evaluation of bilinear alternation

We compare the performance of bilinear alternation, CG and LM for minimizing the objective function in (5.7). The alternation approach chosen is the motion-structure iterations as described in the chapter, where (5.7) is minimized with respect to the in-plane motion parameters only. We analyze the results of the minimization using the three approaches for various levels of out-of-plane motion error. Conventional wisdom suggests that second-order newton methods perform best for the matrix factorization problem [74]; however the results of this experiment for the case of low out-of-plane motion error are surprising because they suggest that bilinear alternation performs fastest under the assumed operating conditions.

A reconstruction problem is obtained by generating 50 random world points and 10 cameras as described above, with 96.4% of the image feature measurements known. The ground plane normal vector at each camera location is perturbed such that the new vector makes angles of $(0^\circ, 3^\circ, 6^\circ, 9^\circ, 12^\circ)$ with the ground-truth vector. The camera centers are

perturbed along the Z-axis of the WCS by errors of $(0\%, 3\%, 5.2\%, 13\%, 26\%, 39\%, 52.1\%)$. For each choice of errors in out-of-plane motion components, we compare the performance of the three algorithms on 500 runs. The in-plane motion components are perturbed by a 100% error in the $X - Y$ locations and a 90° error in the in-plane rotation angle to obtain the starting point for the three approaches. The total number of runs of the minimization for all cases was 17500.

Let the final objective function error value of alternation, CG and LM after convergence be denoted by a^*, c^* and l^* respectively. We have the following mutually exclusive possibilities for each run: (1) $a^* = c^* = l^*$, (2) $a^* = c^* < l^*$, (3) $c^* = l^* < a^*$, (4) $a^* = l^* < c^*$, (5) $a^* < \min(c^*, l^*)$, (6) $c^* < \min(a^*, l^*)$, and (7) $l^* < \min(a^*, c^*)$. The equality in the above expressions is understood to be within a small margin of tolerance. Table 5.2 displays the percentage of runs falling into the above cases for each unique choice of out-of-plane error.

The percentage of runs for which bilinear alternation performs suboptimally compared to LM or CG (i.e. with higher error than the best of CG or LM) is listed in Table 5.3 for each choice of out-of-plane motion errors. This suggests that for moderate levels of errors in the out-of-plane motion, alternation turns out to be a suboptimal choice of algorithm in roughly 17% of cases. At low to moderate errors where we propose the use of our algorithm, alternation turns out to be suboptimal in roughly 9% of cases. In practice, since we switch between in-plane and out-of-plane iterations, the effect of suboptimality in 9% of the cases does not adversely affect the structure and motion reconstruction.

We compared the number of iterations and times taken for each algorithm to reach convergence, where the error function decreases by less than 0.00001%. We find that

Height error	Ground plane vector error														
	0°			3°			6°			9°			12°		
0%		96.8			96.4			94.6			96.6			95.6	
	0	0	0	0	0	0	0	0	0	0	1	0	0.2	4	0
	3.2	0	0	3.6	0	0	5.4	0	0	2.2	0.2	0	0.2	0	0
3%		95.8			95.2			95.2			95.6			93.4	
	0	0.2	0	0.2	0.4	0	0	0.6	0.4	0	0.6	0	0	5.2	0
	4	0	0	4.2	0	0	3.8	0	0	3.8	0	0	0.6	0.6	0.2
5.2%		86.6			87.4			87.2			88.0			84.8	
	0.4	5.2	0.2	0	5.0	0.6	0.8	3.2	0	0	5.4	0.2	0.4	8	0.6
	4.2	2.2	1.2	3.2	2.8	1	3.6	2.2	3	2.8	2.4	1.2	1.4	3.2	1.6
13%		70.8			71.0			71.4			72.6			74.6	
	1.2	9	4.6	2.4	11.2	2.8	1.4	10.2	2.6	1.4	10.4	3.0	1.2	12	2.0
	5.8	6.2	2.4	5.0	3.6	4	5.8	4.4	4.2	5.2	4.2	3.2	2.2	3.6	4.4
26%		71.2			71.2			69.6			68.4			71.4	
	2.6	9.0	2.8	2.0	11.4	2.4	1.4	11.8	4.6	2.6	12.0	3.6	2.2	10	3.4
	3.6	5.8	5.0	3.0	3.8	6.2	2.2	5.2	5.2	4.8	3.0	5.6	3.6	5.4	4
39%		69.4			64.4			64.4			64.4			65.2	
	1.4	12.4	4.6	3.2	14.2	2.6	3.2	11	4	3.2	15.2	3.2	2	12.8	5.4
	4	4.2	4	3.2	5.8	6.6	5	7.8	4.6	2.6	6.2	5.2	6	5.8	2.8
52.1%		60.8			58.4			59.2			56.4			61.8	
	4.4	17	3	5.2	20.8	3.8	5.8	15.4	2.8	5.2	20.2	3	4.8	15.8	3.2
	4.4	6.2	4.2	2.8	4.8	4.2	4.4	5.6	6.8	5	6	4.2	7	3.8	3.6
Layout	(1)														
	(2)	(3)	(4)												
	(5)	(6)	(7)												

Table 5.2: Comparison of performance of Alternation, CG and LM for minimizing the bilinear system in (5.7). Let the final error value of alternation, CG and LM be denoted by a^*, c^* and l^* respectively. We have the following mutually exclusive possibilities for each run: (1) $a^* = c^* = l^*$, (2) $a^* = c^* < l^*$, (3) $c^* = l^* < a^*$, (4) $a^* = l^* < c^*$, (5) $a^* < \min(c^*, l^*)$, (6) $c^* < \min(a^*, l^*)$, and (7) $l^* < \min(a^*, c^*)$. Each cell shows the percentages of runs in all the cases (1) through (7). The arrangement of the seven numbers within each cell is specified by the layout on the last row of the table.

Height error	Ground plane vector error				
	$0°$	$3°$	$6°$	$9°$	$12°$
0%	0	0	0	1.2	4
3%	0.2	0.4	0.6	0.6	6
5.2%	8.6	8.8	8.4	9	12.8
13%	17.6	18.8	18.8	17.8	20
26%	19.8	21.4	22.2	20.6	19.4
39%	20.6	26.6	23.4	26.6	21.4
52.1%	27.4	29.8	27.8	30.4	23.2

Table 5.3: Percentage of runs on which the solution of Alternation was higher than the minimum of the solutions of CG and LM.

alternation, CG and LM take an average time of 0.0257, 0.0550 and 2.8875 seconds respectively. In other words, CG takes 2.1424 times longer, and LM takes 112.5413 times longer than alternation to converge. The corresponding average number of iterations is 48.4882, 316.8153, and 96.5334 (although number of iterations do not directly compare). The large time taken by LM may be attributed to the large Hessian that must be inverted at each iteration. Using a sparse implementation of LM (similar to SBA), we may be able to reduce the computation time. However, we did not implement the sparse LM since in a later experiment we demonstrate that the overall FBSfM algorithm took lesser time-to-converge compared to the SBA implementation. Figure 5.3 shows convergence plots of the three algorithms for $(13\%, 6°)$ error in out-of-plane motion. The red curves corresponding to bilinear alternation are largely below the blue CG and black LM curves.

In this plot, a lower curve implies a faster convergence rate. The plots show the same trend for other choices of error levels. Figure 5.4 plots the convergence curves of LM on a separate figure so that the time axis can be chosen to be in linear scale.

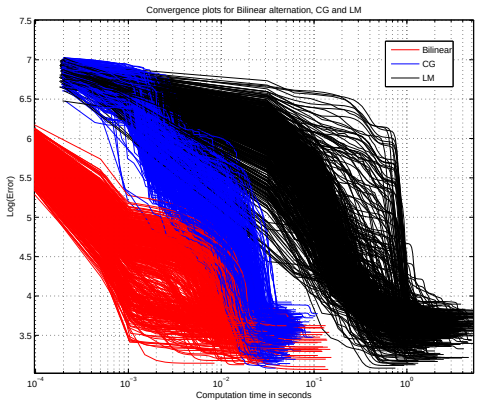
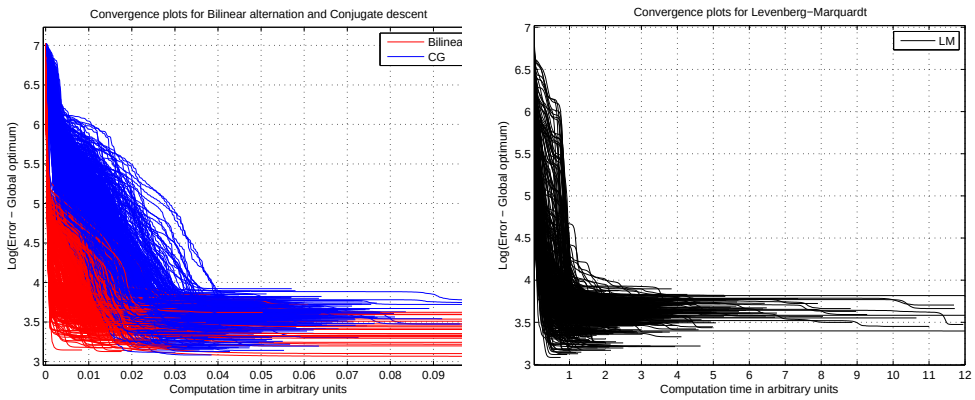


Figure 5.3: This figure illustrates convergence curves plotting the log of error versus computation time for the minimization of (5.7), using bilinear alternation, CG and LM. The plots for bilinear alternation are superior to the other two in terms of convergence rate because the bundle of red curves are largely below the blue and black curves. The reconstruction problem used for these plots involved 10 cameras and 50 features. We used a perturbation of $(3\%, 6^\circ)$ for the out-of-plane translation and ground plane normal angle error respectively.

The strong advantage in computation time along with the fact that alternation converges to the same solution as CG and LM under low out-of-plane noise levels justifies its use under the operating conditions of the dissertation (low error in out-of-plane motion).

5.4.5 Comparison of FBSfM with SBA

We generated a reconstruction problem with 10 cameras and 50 feature points and 96% of the image measurements known. Two sets of initial camera motion solutions were generated for different choices of error levels in motion components. Set 1 had perturbations of $(12\%, 25^\circ, 2.7\%, 2^\circ)$ in the in-plane translation, in-plane angle, verti-



(a) (b)

Figure 5.4: This figure illustrates convergence curves plotting the log of error versus computation time for the minimization of (5.7). Figure 5.4(a) plots the curves for Bilinear alternation and CG, and figure 5.4(b) shows the plots for LM. The x axis in both plots are at the same scale which allows for comparison of LM with the other two algorithms. The plots for bilinear alternation are superior to the other two in terms of convergence rate because the bundle of red curves are largely below the blue and black curves. The reconstruction problem used for these plots involved 10 cameras and 50 feature points. We used a perturbation of $(3\%, 6^\circ)$ for the out-of-plane translation and ground plane normal angle error respectively.

cal position and ground plane normal vector respectively. Set 2 had perturbations of $(20\%, 35^\circ, 0.5\%, 5^\circ)$. Each set had 900 different initial solutions. The structure initializations were obtained by triangulating each 3D point using image feature locations and initial camera parameters, after solving the linear system (5.10) of a structure iteration.

In set 1, minimum error reconstruction produced by SBA was 0.3199 and that produced by our algorithm was 0.3570. In 86% of the runs, the SBA reconstruction error was lower than the minimum error produced by our algorithm. The slightly higher error of our algorithm on successful runs compared to SBA is because we minimize the algebraic as opposed to the reprojection error. On the remaining 14% of the runs, SBA produced an error of above 5. In comparison, our algorithm produced an error lower than 0.4 on 99% of the runs. This illustrates the reliability of our algorithm in producing good reconstruc-

tions when initialized from a number of random initial solutions. In set 2, 38% of the runs had a lower reprojection error for SBA than for FBSfM. The distribution of reprojection errors of both algorithms on each of the sets is illustrated in figure 5.5. The red curves are for set 1 and the blue curves for set 2. The figure plots the cumulative distributions of the reprojection error, hence a higher curve implies a better performing algorithm. In both sets, the curve for FBSfM is largely above that for SBA indicating better performance.

We repeated the experiments on a reconstruction problem with 45 cameras and 250 feature points, with 96.4% measurements known. We obtained 900 initial starting points by perturbing with motion error of $(30\%, 25^\circ, 3.75\%, 4^\circ)$. The initial reprojection errors ranged from 199.4611 to 3476.6 and the SBA code reported failures in minimization on 70.6% of the runs. On the remaining 29.4% of the runs on which SBA succeeded, our algorithm produced reconstructions whose final reprojection error ranged from 0.6277 to 0.7603. SBA produced an error of 0.3723 on 87.5% of the succeeded runs, and errors ranging from 9.0526 to 278.6074 on the remaining 12.5% of the succeeded runs. On the 70.6% of the original runs on which SBA reported failures, our algorithm produced reconstructions ranging from 0.6209 to 0.7848, whereas the errors of SBA ranged from 208.28 to $1.0993e + 05$ (which were very close to the initial errors). These experimental results clearly demonstrate the advantage of our algorithm over SBA, because it generally avoids getting stuck in poor local minima and is more consistent in its results.

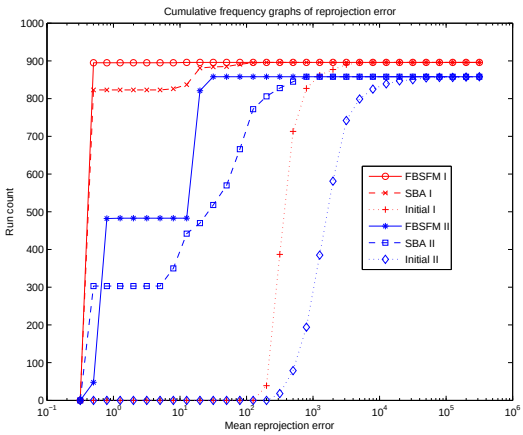


Figure 5.5: This figure shows the cumulative frequency graphs of the reprojection error for SBA and FBSfM along with the error distribution of the initial solutions. The problem involved 10 cameras and 50 feature points. The red curves show the results for set 1 with perturbations of $(12\%, 25^\circ, 2.7\%, 2^\circ)$ in the in-plane translation, in-plane rotaion angle, out-of-plane translation and ground plane normal angle error respectively. The blue curves show the results for set 2 with initial motion error of $(20\%, 35^\circ, 0.5\%, 5^\circ)$. In both sets, the graph for FBSfM is above that of SBA indicating better performance.

5.4.6 Comparison of convergence rates of FBSfM and SBA

We compare the convergence rates of our algorithm with SBA on a reconstruction problem with 300 cameras and 350 feature points, with 62% of the image measurements known. We generated 350 initial solutions by perturbing the ground truth motion with a 3.33% error in the in-plane translation, a 1% error in out-of-plane translation, a 15° error in the in-plane rotation angle and a 4° error in the ground plane vector. SBA converged successfully in 157 of the 350 runs, among which 138 converged to the global minimum, with a reprojection error of 1.1316. On these 138 runs, FBSfM converged to solutions with errors ranging from 1.1427 to 1.1432. Convergence of FBSfM was declared if the number of iterations exceeded 100 or if the error decreased by less than $5e-5$ or 0.0001% whichever was higher. We attribute the slightly higher reprojection error of FBSfM to the fact that it minimizes the algebraic error as opposed to the actual reprojection error.

Since the final errors of FBSfM are very close to those of SBA, we believe this is very close to the global optimum. For the 138 runs described earlier, the number of iterations taken by FBSfM ranged from 93 to 102 with an average of 96.6 iterations and the total computation time ranged from 203.3 to 254.2 seconds with an average of 223.6 seconds. It is possible that because of the termination conditions, FBSfM had not converged to its global optimum in a strict sense but had exhibited flatlining beyond 100 iterations. However, for practical purposes, this is not consequential since a minimum reprojection error solution can be realized by directly minimizing the reprojection error using SBA. Accumulated over all the runs, the amount of time taken for the out-of-plane motion refinement iterations was 49.5% of the total time spent. This fraction ranged from 46% to 55.4% for all the runs.

Sparse BA required iterations ranging from 45 to 4639 for convergence, with an average number of iterations of 484.3. Convergence was declared if the norm of the update to the parameter vector was less than $1E-12$. The total computation time taken ranged from 157 seconds to 17864 seconds with 132 of the 138 runs requiring computation time larger than the maximum time taken by FBSfM on all the runs.

Figure 5.6 shows the convergence plots for FBSfM (plotted in blue) and SBA (plotted in red). SBA exhibits fast convergence when the solution is very close to the global minimum. On the other hand, FBSfM exhibits slow convergence close to the minimum. Figure 5.7 plots the convergence curves of FBSfM and SBA in separate curves on linear scale in the time axis. Although FBSfM is faster than SBA overall, these two algorithms are good candidates to be used together in a hybrid approach. It must be noted that this advantage in computation time is larger as the problem size increases (corresponding to

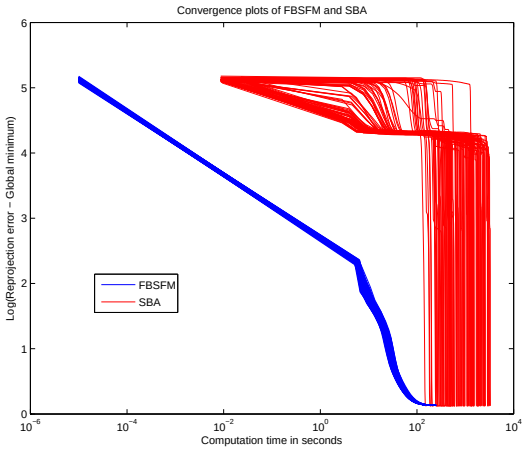
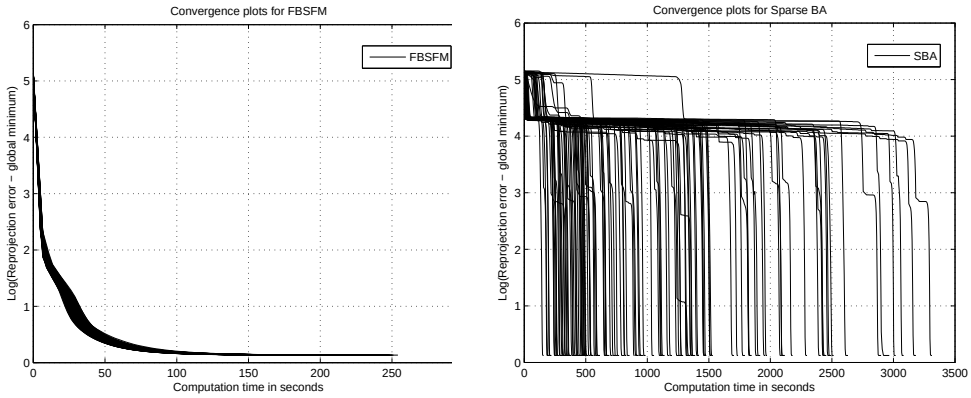


Figure 5.6: This figure illustrates the convergence curves plotting the log of the reprojection error versus computation time for FBSfM and SBA on 138 runs. The blue curves are for FBSfM and the red ones are for SBA. Note that the blue curves are clearly below the red ones indicating that FBSfM converges faster compared to SBA in this experiment.



(a) (b)

Figure 5.7: This figure illustrates the convergence curves plotting the log of the reprojection error versus computation time for the proposed algorithm and SBA. Figure 5.7(a) plots the curves for FBSfM and figure 5.7(b) plots the curves for Sparse BA.

the size of the matrix that needs to be inverted). We have observed that for 300 cameras, FBSfM is much faster than SBA, and for 10 cameras, SBA is faster. The critical problem size above which the reduced Hessian matrix inversion or the slowness of gradient descent starts to become a disadvantage for SBA depends on the machine and the processing resources available. However, for mobile devices, we expect the memory limitation to be the bottleneck where FBSfM would find the most advantageous use (with a low critical problem size).

5.4.7 Comparison with Linear multiview reconstruction and SBA

The results of simulation experiments described here show that FBSfM performs very consistently and advantageously over a wide range of noise level in image feature locations compared to Linear multiview reconstruction [51] and SBA [38].

In order to apply our algorithm, we generate points both on and off an imaginary plane. We generated camera tracks by ensuring that there was sufficient variation in the 'look-angle' of the camera with respect to the imaginary plane, and also sufficient variation in the rotations and camera heights. In the above, 'look-angle' refers to the angle made by the camera principal axis with the plane. We generated the feature tracks by backprojecting the points onto each camera. We add different levels of noise to the generated image feature point tracks and then use them for solving for structure and motion using the various techniques.

In this set of experiments, the additional information for our algorithm is derived directly from the feature point correspondences. Using the feature correspondences known

to lie on the plane, we first compute the induced homography between the various virtual images of the synthetic sequence. Then, we decompose these homography matrices and obtain estimates of the ground plane normal and heights that provide the side information for algorithm [71]. The decomposition process requires knowledge of the calibration matrix of the camera, and is described in detail in the appendix B. The estimates of GPN and height are inaccurate when we have errors in homography estimates (due to errors in image features) or errors in the calibration matrix. Hence, we report results under two scenarios: when we have (a) accurate camera calibration and (b) inaccurate camera calibration.

5.4.7.1 Results with accurate calibration

Figs. (5.8(a),5.8(b)) illustrate the mean reprojection error, and the deviation of these reconstruction estimates at a range of noise levels for the three techniques. The ground truth camera calibration was used in this case. Over a wide range of noise levels, we see that FBSfM performs consistently and competitively. At very low noise levels, Rother's performs slightly better than FBSfM. But at realistic and high pixel noise levels, FBSfM seems to perform the best.

The experiment was repeated for 70 trials under scenarios with different pixel noise levels. Fig. (5.10(a),5.10(b)) plot the reconstruction errors for each method for all the trials. We observe that in different settings, both bundle adjustment (BA) and Rother's give bad solutions in many cases. BA is considered the gold standard in SfM algorithms because it minimizes the reprojection error starting with a good initial solution. But it

involves a highly nonlinear iterative refinement and is prone to getting stuck at a local minimum. The mean reprojection error of BA varies widely indicating that it may be affected by local minima unlike FBSfM which is a lot more consistent.

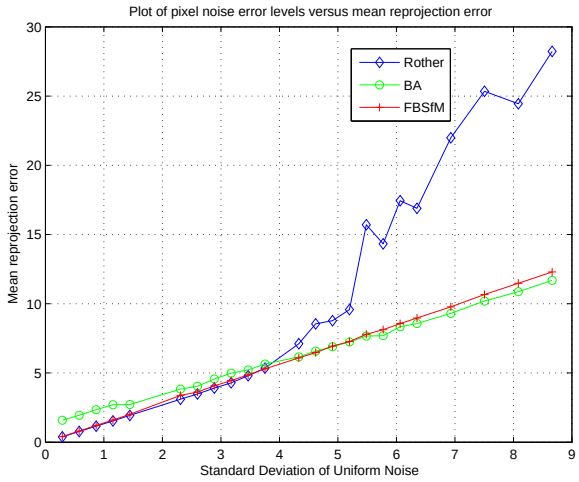
5.4.7.2 Results with inaccurate calibration

The above experiments were repeated for the case where we had large calibration errors (around 20% error in the focal lengths and camera center estimates). The mean reprojection error plots (Fig. 5.9(a)) are similar, and the deviations of the estimates are lowest for FBSfM (Fig. 5.9(b)).

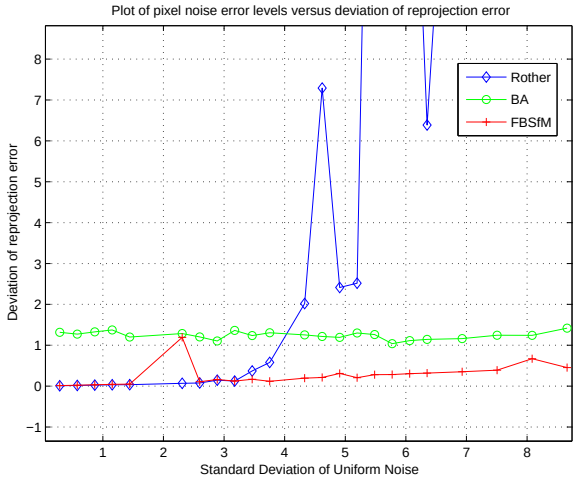
The experiment was repeated 70 times under scenarios with different pixel noise levels with an inaccurate calibration matrix as depicted in Fig. (5.11(a),5.11(b)). The observations from these figures is similar to the case of accurate calibration.

5.4.8 A note on alternation methods

A popular alternative to SBA that falls within the class of alternation algorithms is resection-intersection [38]. This involves iterations where (1) the cameras are fixed and structure variables are updated, and (2) the structure is fixed and camera matrices are updated. Each iteration is of low complexity similar to our bilinear alternation, but it involves non-linear minimization in all the iterations. The key difference of our work is the use of additional measurements for the decomposition of motion parameters that leads to a better performing algorithm for low out-of-plane noise conditions. Our experimental results do not show a clear advantage over either SBA or resection-intersection from a

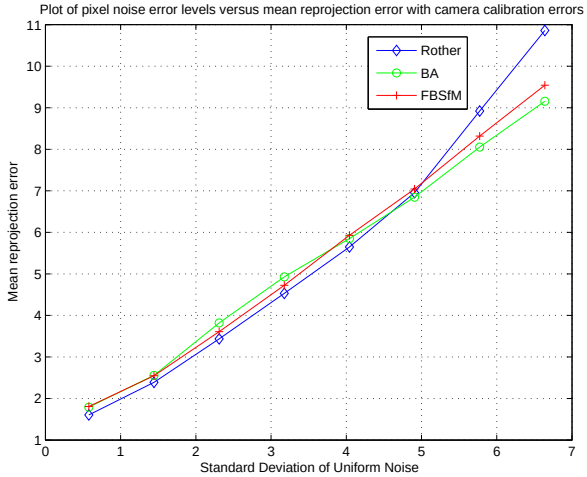


(a) Mean reprojection error

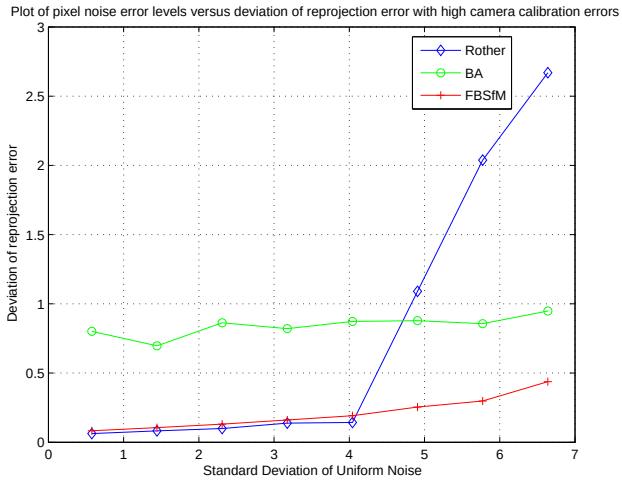


(b) Variance of reprojection error

Figure 5.8: These plots show the statistics of the reprojection errors of the reconstructions on synthetic data at various noise levels in the image feature points. They show the mean and variance of the estimates for three techniques: FBSfM, Rother's and Bundle Adjustment. The ground truth calibration matrix was used in the process of extracting initial estimates of GPN and height (additional information) from the homographies. Initial values for the GPN and height differed from the ground truth because the homography estimates were obtained from noisy feature correspondences. There was low-to-moderate error in the initial GPN and height estimates. In addition, the iterative techniques (FBSfM and BA) used the same initial solutions to start the update iterations.

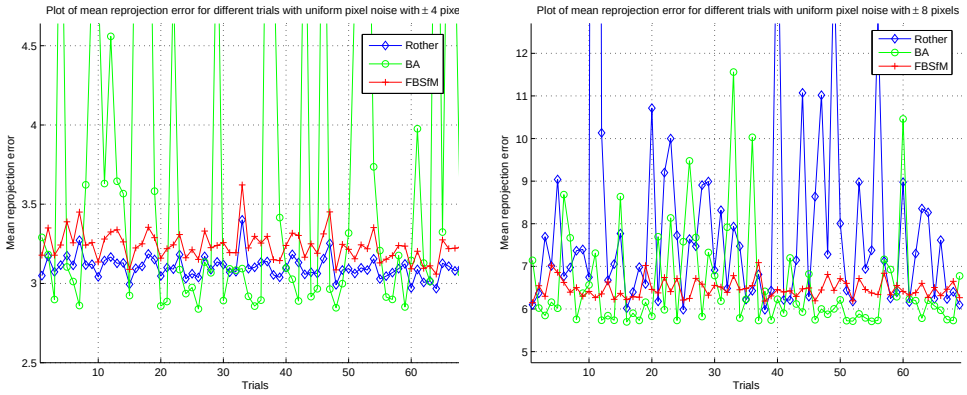


(a) Mean reprojection error



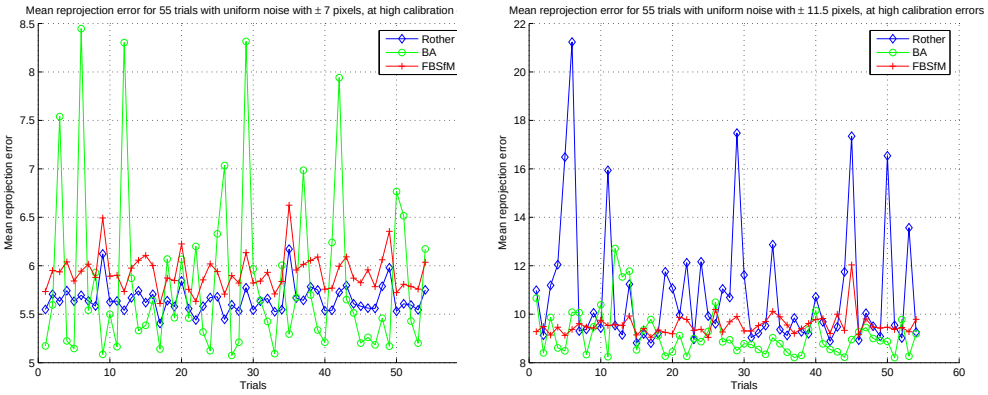
(b) Variance of reprojection error

Figure 5.9: These plots show the statistics of the reprojection errors of the reconstructions on synthetic data at various noise levels in the image feature points. They show the mean and variance of the estimates for three techniques: FBSfM, Rother's and Bundle Adjustment. There was a large error (from ground truth) in the calibration matrix that was used in the process of extracting initial estimates of GPN and height (additional information) from the homographies. Hence, this experiment serves as a case where evaluation was performed with high levels of error in the GPN and heights (from ground truth). FBSfM and BA were started with the same initial solution for the reconstruction.



(a) Noise level of ± 4 in image features (b) Noise level of ± 8 in image features

Figure 5.10: These plots illustrate the reprojection errors of the reconstructions on synthetic data. The reconstruction errors are plotted for 70 different trials. There is a uniformly distributed noise level of ± 4 pixels in the image features for the plot 5.10(a) and a noise level of ± 8 pixels for the plot 5.10(b). They serve to give an idea of how the various methods perform for repeated trials of the same experiment. Results for three different techniques are shown: FBSfM, Rother's and Bundle Adjustment. The ground truth calibration matrix was used in the process of extracting initial estimates of GPN and height (additional information) from the homographies. Hence, this experiment serves as a case where evaluation was performed with low to moderate levels of error in the GPN and heights.



(a) Noise level of ± 7 in image features (b) Noise level of ± 11.5 in image features

Figure 5.11: These plots illustrate the reprojection errors of the reconstructions on synthetic data. The reconstruction errors are plotted for 70 different trials. There is a uniformly distributed noise level of ± 7 pixels in the image features for Fig. 5.11(a) and a noise level of ± 11.5 pixels for Fig. 5.11(b). They serve to give an idea of how the various methods perform for repeated trials of the same experiment. Results for three different techniques are shown: FBSfM, Rother's and Bundle Adjustment. There was a large error (from ground truth) in the calibration matrix that was used in the process of extracting initial estimates of GPN and height (additional information) from the homographies. Hence, this experiment serves as a case where evaluation was performed with high levels of error in the GPN and heights.

general class of initial solutions. However, the advantage shows up very clearly when the out-of-plane motion is known (with low error) through sensor measurements. To the best of our knowledge, we do not know of a previously proposed variant of resection-intersection for the specific setting discussed here. Earlier works [38] have evaluated the most general variant of resection-intersection and have found it to be suboptimal compared to SBA in terms of accuracy. Based on this study, we conclude that the cumulative frequency graphs of resection-intersection are expected to be below that of SBA.

5.5 Experiments

5.5.1 Experiments on an Indoor Handheld Sequence

We report reconstruction results on an indoor image sequence taken with a digital camera of a toy car resting on a plane. SIFT features were extracted and matched across the image sequence. Inter-image homographies were estimated robustly using RANSAC, and planar points were separated from those off the plane. The homographies were decomposed as described in appendix B to obtain the additional information. The reprojection errors for the three techniques FBSfM, Rothers' and BA are shown in Table 5.4. Figure 5.12 illustrates the texture mapped 3D model of the car. This is an example of a short video sequence, with the number of keyframes being 13 and the number of feature points around 60.

	Reprojection error
FBSfM	2.79
Rother	3.40
BA	4.10

Table 5.4: Mean reprojection error of the reconstructions of FBSfM, BA, and Rother's for the indoor toycar sequence.

Figure 5.12: This figure illustrates three views of the texture mapped 3D model of the car in the toycar sequence, obtained by interpolating from sparse structure estimates generated by FBSfM. Manually assisted feature point matches were also used to generate this result, to ensure the display of a full 3D model.

5.5.2 SfM on StreetView data

The Google StreetView Research dataset consists of roadside image sequences and metadata with the locations and orientations of the camera corresponding to each image, solved using GPS and IMU measurements deployed onboard. The metadata contained the additional measurements in the form as required by the proposed algorithm. We chose a segment of the dataset containing 150 images when the car is moving on a single road. We obtained feature tracks by SIFT feature detection and matching, and used RANSAC and the epipolar constraint to prune out outliers. After these post-processing steps, we obtained 3145 distinct feature tracks in the sequence.

We obtained an initial solution for the camera motion by assuming a straight line motion and placing the camera centers equally distributed on the line. We initialized the out-of-plane camera rotation matrices using the direction vectors obtained from the metadata, and the in-plane rotations were all fixed to zero. The heights of the cameras were later fixed to the measurements obtained from the metadata. This provided an initial solution for the camera motion. We used this initial motion and the feature trajectories to triangulate the 3D points using structure iteration equations (5.10). The triangulated $3D$ locations were used as the initial structure solution. Both FBSfM and SBA were initialized with the same starting point. The initial reprojection error was 237.49. FBSfM converged to a solution with an error of 1.99 and SBA converged to a solution of 111.88. Figure 5.13 illustrates the top view of the reconstructed 3D points and the camera path corresponding to the FBSfM solution. When we initialized SBA with the final solution of FBSfM, the reprojection error reduced to 1.342.

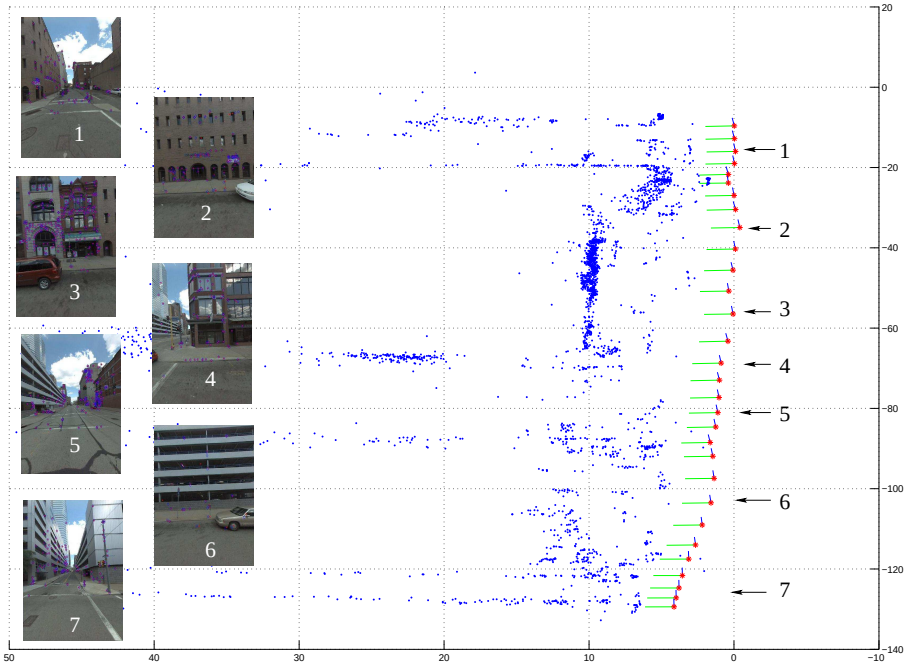


Figure 5.13: This figure shows the top view of the reconstructed 3D points and the camera path obtained by solving for structure and motion from 150 images of the streetview sequence using the proposed algorithm. The red points show the camera centers and the green lines show the optical axis at each camera location. A few images in the sequence are shown in the left of the figure. We can clearly distinguish the three intersecting roads in the reconstruction, with the road in the middle approximately twice as wide as the other two roads.

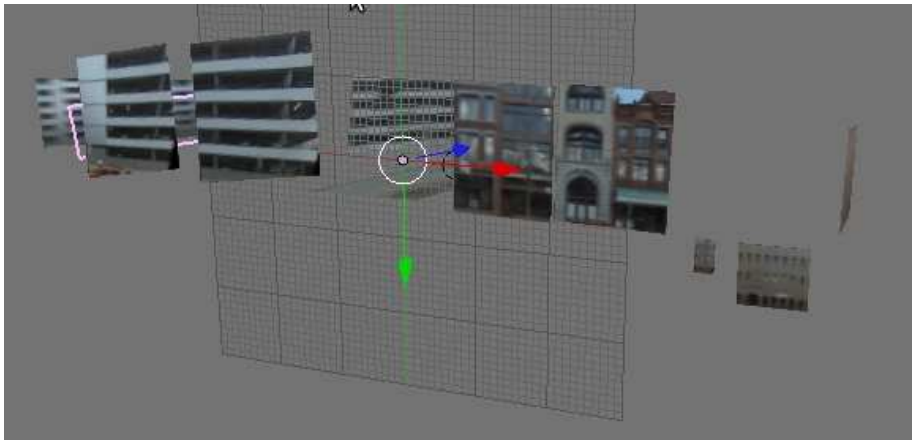


Figure 5.14: This figure shows a texture mapped 3D model of the scene imaged in the StreetView sequence. Since the urban scene consists primarily of buildings and other man-made structures, we fit several planes to the reconstructed 3D points. The textures for these planar patches were obtained from the corresponding images, and these textures were applied to the planar patches using the Blender 3D modeling tool. Novel views of the texture-mapped model were rendered using the same tool.

Figure 5.14 illustrates the textured 3D model of the sequence generated using the Blender modeling tool. Figure 5.15 illustrates novel views synthesized using the textured 3D model of figure 5.14 obtained using Blender software. Figure 5.16 illustrates the trajectory of the camera overlaid on a map in Google Earth software.

Since we do not currently use sparse representations for measurement matrices, our implementation was not suited to execute multiple trials on the earlier sequence for measuring computation times. Hence we select a subsequence with lesser number of feature points. From the SfM solution with error 1.342, we selected the first 70 frames, and 755 points with reprojection error less than 3 for all frames. We used SBA for the 755 point subsequence and obtained an SfM reconstruction with error of 0.4013. We repeated the camera configuration of the 70 cameras twice by translating in the Y and Z directions by chosen distances. This produced a total of 210 cameras. We generated the feature points on the virtual cameras by reprojecting the original 3D points on each of the images

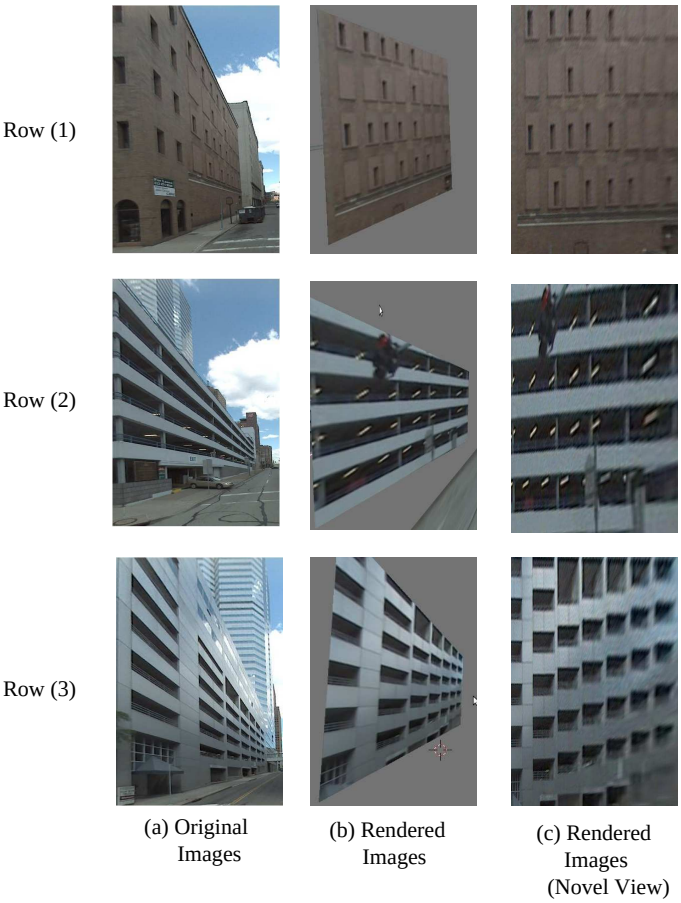


Figure 5.15: This figure illustrates three different parts of the StreetView scene. Each row shows one particular facade present in the scene. The images in column (a) show the original images from the sequence. The images in column (b) show the synthesized images generated using the 3D model from a viewpoint that was similar to (a). The images in column (c) show the synthesized images generated using the 3D model from a novel viewpoint not present in the original sequence.

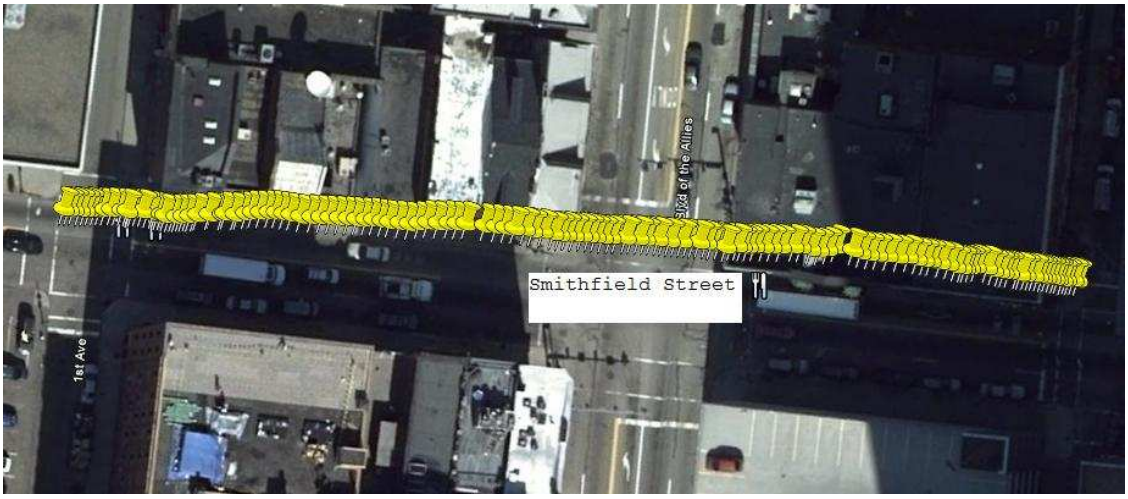
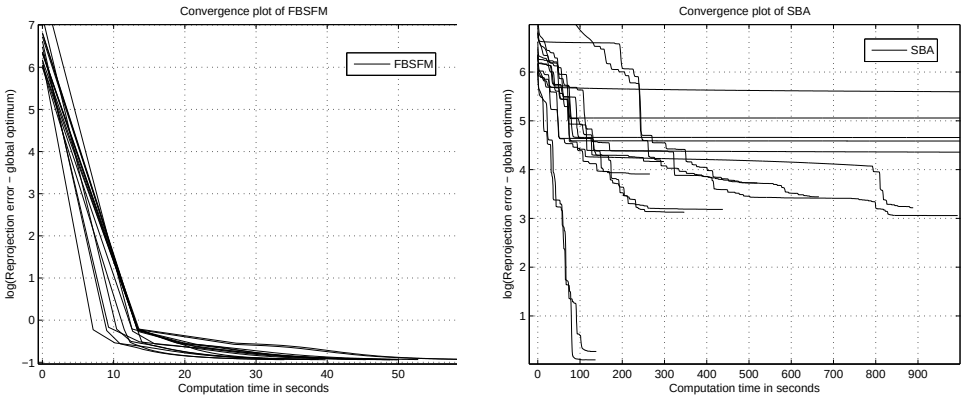


Figure 5.16: This figure shows the trajectory of the camera in the StreetView sequence obtained by solving for the SfM using our algorithm. The recovered trajectory was converted fro UTM coordinates to Latitude-longitude and overlaid on the map in Google Earth. The trajectory reflects the path taken by the camera as inferred from the image sequences. The camera moves on the 'Smithfield Street' in downtown Pittsburg. It starts from the intersection of Smithfield and 1^{st} avenue, goes past the intersection with 'Blvd of the Allies', and ends after the intersection with 3^{rd} avenue.

and adding random noise to the pixel locations, to simulate feature detection errors. We use this SfM problem with 210 cameras, 755 points with 10% of the measurement matrix known, as the test problem for measuring computation times. The ground truth reprojection error (global optimum) for this problem was 0.3892. We generated initial solutions by perturbing the in-plane translation by 11%, out-of-plane translation by 1%, in-plane rotation angle by 7° and direction vector by 1°. Among the 15 runs of SBA that terminated successfully, the final reprojection errors of two best runs were 1.104 and 1.311, and the rest of the errors were all above 20. In contrast, the final errors of FBSfM ranged from 0.3906 to 0.3948 with the motion and structure reconstructions very close to the ground truth. The average time taken by the proposed algorithm to reach a solution was 45.957 seconds. Convergence of our algorithm was declared if the iteration count reached 100 or if the reprojection error decreased by less than $5e-5$. All trials used 100 iterations. The

algorithm probably exhibit flatlining behavior beyond 100 iterations since the error was decreasing very slowly. However, beyond this point, SBA is a better choice for minimizing the reprojection error and can be used in a hybrid approach along with FBSfM. We do not estimate the average time taken by SBA since none of the runs converged close to the global optimum. However, the time taken for the two best runs were 136.7 and 138.5 seconds. Figure 5.18 plots the convergence curves for both algorithms. The bundle of blue curves for FBSfM is clearly below the red ones of SBA indicating a faster overall convergence rate. The curves suggest that towards the start of the minimization, FBSfM converges faster but when the solution is close to the minima, SBA converges faster. Figure 5.17 plots the convergence curves for both algorithms in separate graphs with the time axis in linear scale.



(a) (b)

Figure 5.17: This figure illustrates the convergence curves plotting the log of the reprojection error versus computation time for the proposed algorithm and SBA. Figure 5.7(a) plots the curves for FBSfM, and figure 5.7(b) plots the curves for Sparse BA. Figure 5.7(b) cuts off the time axis at 1000 sec, however the maximum time taken was 2140.

These results illustrate the computational advantages of FBSfM over SBA and make it a good candidate for urban modeling which involves solving large reconstruction prob-

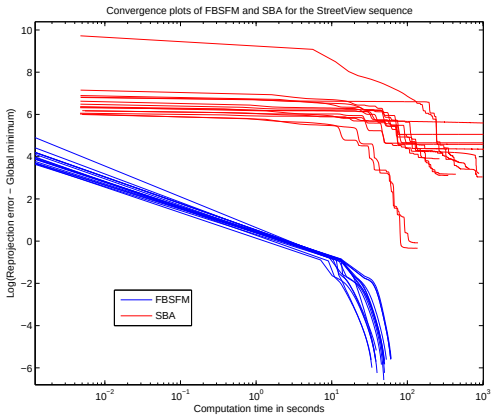


Figure 5.18: This figure illustrates the convergence curves plotting the log of the reprojection error versus computation time for FBSfM and SBA. The blue curves are for FBSfM and the red ones are for SBA. The blue curves are clearly below the red ones indicating that FBSfM converges faster compared to SBA in this experiment.

lems.

5.5.3 SfM on Static Points in Aerial Video - 1

The algorithm was tested on a long real aerial video sequence. The metadata associated with the sequence made available the normal vector and the camera heights from additional sensors on-board. This information was computed from the measurements of the slant range and the azimuth, elevation and twist of the camera (which was part of the metadata). Around 1700 feature points were tracked through the sequence, which was 225 frames long. Around 30 keyframes were selected, which were the frames for which timestamped metadata was available. This sequence is a dense reconstruction problem because a large percentage of the feature points is observed in each view. Two versions of the algorithm were compared, one with the initial GPN and Tz estimates obtained from the homographies, and another with estimates obtained from the metadata.

From table 5.5, we infer that FBSfM produces the best reconstruction when used

with the metadata. With the homography-based estimates of GPN, the reprojection error was worse, and this was to a largely due to a high reprojection error in one particular frame. We feel this is because of a bad initial estimate of the GPNs and heights. The homography decomposition step is sensitive to the calibration matrix. Rother's method requires the solution of a very large matrix equation which is highly time and memory consuming. Hence we could perform a reconstruction for only a few of the points. Bundle adjustment requires the inversion of a large Hessian matrix. Our technique performs faster than these techniques because at any iteration, the peak memory requirement is very limited.

FBSfM with metadata	FBSfM	Rother	BA
2.83	10.5	24.32	5.56

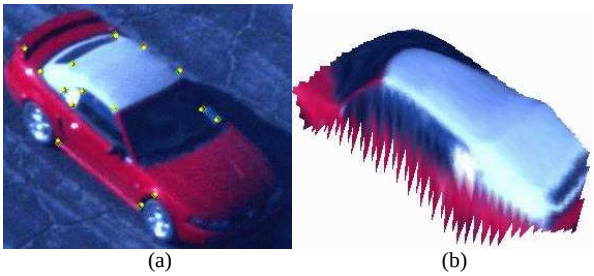
Table 5.5: Mean reprojection error for the long aerial video sequence in which we had 225 frames and 1700 features.

5.5.4 SfM on Moving Objects in Aerial Video - 2

We report experiments on reconstruction of moving objects on a planar scene. The theoretical derivations suggest that the moving object is assumed stationary, and there is a separate camera moving over it. We did not compute the inter-image homographies, but obtained the GPN and height from the metadata, and estimate the structure and in-plane motion using the bilinear algorithm without direction vector or height refinement.

Fig. (5.19a) shows a snapshot of a moving vehicle with detected and reprojected features shown in yellow dots and green squares respectively. Fig. (5.19b) shows the

reconstructed 3D-model. The reprojection error was around 0.7 pixels. This experiment illustrates how we can use our technique to perform SfM on objects moving on a plane, when these objects are rotating and translating on the plane, in a general way. Earlier techniques [53, 54] assume constrained object motion in order to compute the structure.



(a) (b)

Figure 5.19: Fig (5.19a) shows a snapshot of a moving car in the video sequence, with the detected features shown in yellow dots. It also shows the reprojected features shown as green squares. Fig. (5.19b) shows the partially reconstructed 3D model of the car.

5.6 Discussion and Conclusions

We discussed the importance of exploiting the available inertial measurements in the SfM estimation framework. We described a fast, robust and scalable SfM algorithm that leverages additional measurements for computing the scene structure and camera motion from a sequence of images. We described how this algorithm tackles the needs of scalability and speed required in current and future SfM applications involving very large datasets. We show that with the availability of measurements of the gravity vector and camera height, the SfM problem can be simplified into a bilinear form and solved using a fast scalable iterative procedure. The following are the lessons learned from the experiments.

1. FBSFM produces better solutions than SBA from initial solutions with low out-of-

plane motion error. We have compared the performance of both for as much as 5° in ground plane vector error and 5% in the camera vertical position error.

2. The total time taken by FBSFM to attain convergence (with termination conditions set by a maximum number of iterations or minimum decrease in error) is less than that of SBA for large-sized problems when initialized from starting points with low error in out-of-plane motion.

3. When initialized from a large number of random starting points, FBSFM seems to converge to solutions that are lower than those produced by SBA (as is demonstrated by the cumulative frequency graphs).

The chapter provides a direction of research in SfM in the present context where we have large video datasets with associated metadata, or data acquisition platforms with multiple sensors.

One avenue for future research is to develop algorithms that combine video with the raw data from inertial sensors for better SfM algorithms. Inertial measurements give us the accelerations and rotation rates of the cameras which gives us more information about the camera motion than the measurements of gravity vector and height. Such an algorithm will potentially be more generally applicable in scenarios where we have cameras with additional sensors.

Chapter 6

Conclusions and Future Research Directions

6.1 Summary

This dissertation revolves around the present day availability of video datasets with metadata containing additional information from sensors such as inertial measurement units, global positioning systems, wheel speed readers (in ground-based platforms), magnetic compasses etc. The ubiquitous deployment of these additional sensors along with cameras makes it possible to collect such metadata along with video sequences. With additional data, the development of better video processing algorithms leveraging the additional information is of prime research importance. In this dissertation, we propose novel studies and algorithms for several video processing tasks such as stabilization, mosaicking, moving object detection and structure from motion, leveraging the presence of metadata with video datasets. The contributions are summarized as follows.

1. We propose a methodology to reduce the computational cost of intensity-based registration by using a subset of salient pixels with high gradients for the registration. We study the variation of number of iterations and registration accuracy with the number of pixels used for registration.

2. We propose a feature-based algorithm for stabilization and moving object detection in aerial datasets with accompanying metadata. The metadata is useful because

it provides partial information about the camera motion that reduces the motion model to a pure translation model. We propose and demonstrate the working of a joint feature tracking and segmentation algorithm that stabilizes and identifies moving objects in the video sequence

3. We use additional information about camera height and gravity direction to come up with a fast, robust, and scalable algorithm for structure from motion. The proposed algorithm, which we call Fast Bilinear Structure from Motion (FBSfM) is experimentally demonstrated to have favorable properties compared to the sparse bundle adjustment algorithm. In specific scenarios with low error in the gravity and height measurements, we are able to demonstrate that FBSfM is faster than bundle adjustment, and produces more consistent results from a variety of initial solutions. We also describe results on several real datasets including the Google StreetView dataset.

6.2 Future work

There are several avenues for extending the work presented in this dissertation. In chapter 4, we presented an algorithm for joint tracking and segmentation, where we used a pure translation model for tracking the background features jointly. We assumed only one motion model for the background and we classified features as inliers and outliers with respect to the single background model. The outlier features were tracked as independent KLT features. We plan to extend this framework under the assumption of multiple motion models in the image sequence. The use of multiple motion models will extend the appli-

cation of this framework for more general scenes. In addition, though we have used the pure translation model, the algorithm would work just as well if we solved the parameters of a higher order model such as affine or homography. This will extend the applicability of the algorithm for scenes with multiple planes.

In chapter 5, we presented a structure from motion algorithm leveraging the availability of gravity and height measurements obtained from inertial sensors. This leads to a bilinear algorithm that is demonstrated to be fast and robust. However, we do not use the complete information available from these sensors to fuse with image measurements. In particular, fusing image features with rate measurements from a gyro and accelerometer for estimating trajectory can lead to algorithms for a broader range of scenarios beyond what is addressed in this dissertation.

Appendix A

Proofs of convergence

We reproduce the Global Convergence Theorem as stated in [77] for the sake of completeness:

Theorem 1 (GCT): Consider a topological space X, a solution set $\Gamma \subset X$, and let A be an algorithm on X. Suppose that given x_0, a sequence of points $\{x_k\}_{k=0}^{\infty}$ is generated in X such that for $k \geq 0$, $x_{k+1} \in A(x_k)$. Suppose

1. all the points X_k are generated in a compact set $S \in X$,
2. there exists a continuous function $Z : X \rightarrow \mathbb{R}$ such that
   - if $x \notin \Gamma$, then $Z(y) < Z(x)$ for all $y \in A(x)$
   - if $x \in \Gamma$, then $Z(y) \leq Z(x)$ for all $y \in A(x)$
3. the mapping A is closed at points outside Γ.

then the limit of any convergent subsequence of x_k belongs to Γ.

Let S denote the matrix containing the $3D$ coordinates of all the points. Let M_I denote the matrix containing the in-plane motion parameters (t_x, t_y, θ) of all the frames, and M_O contain out-of-plane motion parameters (t_z, ϕ, ψ) for all frames. We show using the Global Convergence Theorem that the proposed FBSFM algorithm converges to a solution (S^*, M_I^*, M_O^*) and the error converges to a local minimum E^*.

Let the set Ω denote the set of all valid structure and motion parameters such that $\|T_z\|^2 = k$, and the angles $(\theta, \phi, \psi) \in [-\pi, \pi]$. This set is a compact set of the parameter space $\mathbb{R}^{3n+6m}$. Each iteration maps the current estimate of (M_I, M_O, S) to the next one computed according to the equations. In case of depth and motion iterations, this involves solving linear systems. In case of side-information refinement iterations, this involves performing gradient descent on the error functions. After each iteration, since the points $x_k = (M_I^k, M_O^k, S^k)$ generated by the iterations belong to S, the first assumption of GCT holds.

Let us define the descent function Z to be the same as the error function E (5.7) being optimized. The solution set $\Gamma \in X$ contains the critical points of the objective function. Since the error is non-increasing, it is easily seen that $E(y) \leq E(x)$ for any x. We have to show that the error is strictly decreasing when we apply the algorithm starting from a point that is not in the solution set Γ. Alternately, we must show that after the iterations, if the error is constant, then we started off from a point in the solution set. This is proved using a very similar reasoning as done in [56, 78]. Hence, the second condition of GCT is also satisfied.

To show that the mapping A is closed, we take an approach similar to [56, 78] and decompose A into elementary mappings corresponding to each iteration:

- A_1 associates with $(S^{(k)}, M_I^{(k)}, M_O^{(k)})$ the solution $S^{(k+1)}$ obtained by solving the depth iterations. This is a continuous point-to-point mapping, since the linear system in (5.9) has a unique solution.

- A_2 associates with $(S^{(k+1)}, M_I^{(k)}, M_O^{(k)})$ the solution $M_I^{(k+1)}$ obtained by solving the

motion iterations. This is a continuous point-to-point mapping, since the equations in (5.15), (5.18) give rise to a unique solution.

- A_3 associates with $(S^{(k+1)}, M_I^{(k+1)}, M_O^{(k)})$ the solution $M_O^{(k+1)}$ obtained by refining the out-of-plane motion parameters using Levenberg-Marquardt (LM) iterations. This is a continuous point-to-point mapping since LM computes its solutions using line-search.

The composition of closed point-to-point mappings is itself closed [56, 77]. Since all the conditions of GCT are satisfied, the FBSFM algorithm is globally convergent.

Appendix B

Decomposition of homographies to obtain additional information

When a dominant plane is present in the scene, we can use the plane normal and the height from the plane as side information in our algorithm. We can accumulate the homographies induced by the plane from multiple views and use a decomposition technique [71] to compute the plane normals and the heights. The inter-image homographies are robustly estimated using RANSAC.

We briefly describe the technique that can be used to estimate the ground plane normals, rotations, translations and other geometric parameters from a set of homographies induced by a plane between multiple views.

Suppose H_{ij} is the homography from view j to view i. It is expressed in terms of rotations, translations and the ground plane normals as $H_{ij} = R_{ij} + \frac{t_{ij} n_j^T}{d_j}$. Here, Q_{ij} denotes quantities from view j to view i, n_j is the ground plane normal at view j, and d_j is the corresponding perpendicular distance from the plane. We can compose a matrix G containing the various homographies as follows:

$$G = \begin{bmatrix} H_{11} & \cdots & H_{1m} \\ \vdots & \ddots & \vdots \\ H_{m1} & \cdots & H_{mm} \end{bmatrix} \tag{B.1}$$

The individual homographies are normalized so that their median singular value is 1. The G matrix can be decomposed as $G = \bar{R} + \bar{T}\bar{N}^T$ where G and $\bar{R}$ are $3m \times 3m$, $\bar{T}$ is $3m \times m$ and $\bar{N}$ is $3m \times m$ with m being the number of views. Here, $\bar{N}$ is a block diagonal matrix

of all the normal vectors, and $\bar{R}, \bar{T}$, are a block matrices with the $(i,j)^{th}$ block composed of R_{ij} and T_{ij} respectively.

It has been shown in [12] that the matrix $L = WG^TW^{-1}$ has rank 3, where

$$W = diag(I_3, det(H_{21})I_3, \ldots det(H_{m1})I_3) \tag{B.2}$$

L has three eigenvectors with eigenvalues of $\lambda = m$. As the vector of ground plane normals $(n_1, n_2, \ldots, n_m)$ can be shown to be an eigenvector of L with eigenvalue of m, it is a linear combination of the top three eigenvectors of L. They elaborate a technique to solve for the normals using these constraints. After recovering the normal vectors, the translations and rotations can be solved using $\bar{T} = G\bar{N} - \bar{N}U_m$ and $\bar{R} = G - G\bar{N}\bar{N}^T + \bar{N}U_m\bar{N}^T$, where U_m is an $m \times m$ matrix of ones. Thus, the heights from the ground plane can be estimated using only the homography matrices and normals using the formula $d_k = \frac{d_1}{n_k^T H_{1k} n_1}$, where d_k is the height of the camera at view k.

Bibliography

[1] "Google image search." [Online]. Available: http://images.google.com/

[2] S.-W. Joo and R. Chellappa, "A multiple-hypothesis approach for multiobject visual tracking," *IEEE Trans. Image Processing*, vol. 16, no. 11, pp. 2849–2854, Nov. 2007.

[3] M. Ramachandran and R. Chellappa, "Stabilization and mosaicing of aerial videos," in *Proc. IEEE Int'l. Conf. on Image Processing*, October 2006, pp. 345–348.

[4] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 1, pp. 133–135, 1981.

[5] O. Faugeras, *Three-dimensional computer vision: A geometric viewpoint*. MIT Press, 1993.

[6] J. Oliensis, "A critique of structure from motion algorithms," NEC Research Institute, Technical Report, 2000.

[7] Y. S. Yao and R. Chellappa, "Selective stabilization of images acquired by unmanned ground vehicles," *IEEE Trans. Robotics and Automation*, vol. 13, no. 5, pp. 693–708, October 1997.

[8] C. Morimoto and R. Chellappa, "Fast 3D stabilization and mosaic construction," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 1998, pp. 660–667.

[9] A. Mitiche, *Computational Analysis of Visual Motion*. Plenum, 1994.

[10] O. Faugeras, *Three-Dimensional Computer Vision*. Cambridge, MA: MIT Press, 1993.

[11] R. Hartley and Z. A., *Multiple View Geometry in computer vision*. Cambridge, UK: Cambridge University Press, 2000.

[12] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *Int'l Journal Computer Vision*, vol. 9, no. 2, pp. 137–154, Nov 1992.

[13] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography," *Commun. Assoc. Comp. Mach.*, vol. 24, pp. 381–395, 1981.

[15] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[16] C. Morimoto and R. Chellappa, "Fast electronic digital image stabilization for off-road navigation," in *Real-time Imaging*, vol. 2, 1996, pp. 285–296.

[17] S. Avidan and A. Shashua, "Threading fundamental matrices," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 73–77, January 2001.

[18] S. Baker and I. Matthews, "Lucas Kanade 20 years on: A unifying framework," *Int'l Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, Feb 2004.

[19] A. Bartoli, "Groupwise geometric and photometric direct image registration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 12, pp. 221–255, Dec 2008.

[20] Y. Keller and A. Averbuch, "Multisensor image registration via implicit similarity," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, May 2006.

[21] M. Ramachandran, A. Veeraraghavan, and R. Chellappa, "Video stabilization and mosaicing," in *The Essential Guide to Video Processing*, 2009, pp. 109–140.

[22] S. Srinivasan and R. Chellappa, "Noise-resilient estimation of optical flow by use of overlapped basis functions," *Journal of the Optical Society of America - A*, vol. 16, no. 3, March 1999.

[23] P. H. S. Torr and A. Zisserman, "Feature-based methods for structure and motion estimation," in *Workshop on Vision Algorithms: Theory and Practice*, 1999.

[24] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1151–1163, Jul 2002.

[25] S. Birchfield and S. Pundlik, "Joint tracking of features and edges," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2008, pp. 1–6.

[26] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 12, pp. 185–203, 1981.

[27] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *Proc. IEEE Int'l Conf. on Computer Vision*, Oct. 2009.

[28] A. Buchanan and A. Fitzgibbon, "Combining local and global motion models for feature point tracking," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, June 2007, pp. 1–8.

[29] M. Irani, P. Anandan, and S. Hsu, "Mosaic based representations of video sequences and their applications," in *International Conference on Computer Vision, Cambridge, MA*, 1995, pp. 605–611.

[30] B. Rousso, S. Peleg, I. Finci, and A. Rav-Acha, "Universal mosaicing using pipe projection," in *International Conference on Computer Vision, Mumbai, India*, 1998, pp. 945–952.

[31] S. Peleg, B. Rousso, A. Rav-Acha, and A. Zomet, "Mosaicing on adaptive manifolds," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1144–1154, October 2000.

[32] J. Wang and E. Adelson, "Representing moving images with layers," *IEEE Trans. on Image Processing*, vol. 3, no. 5, pp. 625–638, September 1994.

[33] R. Kumar, P. Anandan, and K. Hanna, "Shape recovery from multiple views: a parallax based approach," in *ARPA Image Understanding Workshop, Monterey, CA*. 2929 Campus Drive, Suite 260, San Mateo, California 94403: Morgan Kauffmann Publishers, Nov. 1994. [Online]. Available: citeseer.ist.psu.edu/kumar94shape.html

[34] H.-Y. Shum and R. Szeliski, "Construction of panoramic mosaics with global and local alignment," *International Journal of Computer Vision*, vol. 36, no. 2, pp. 101–130, February 2000.

[35] R. Szeliski, *Image Alignment and Stitching: A Tutorial*. Now Publishers Inc., 2006.

[36] S. Peleg and J. Herman, "Panoramic mosaics by manifold projection," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 1997, pp. 338–343.

[37] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[38] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Lecture Notes in Computer Science*, vol. 1883. London, UK: Springer-Verlag, 2000.

[39] K. Madsen, H. Nielsen, and O. Tingleff, *Methods for Non-Linear Least Squares Problems*, 2nd ed. Technical University of Denmark, 2004.

[40] M. I. A. Lourakis and A. A. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Trans. Math. Software*, vol. 36, pp. 1–30, Jan. 2009.

[41] M. Byrod and K. Astrom, "Bundle adjustment using conjugate gradients with multiscale preconditioning," in *Proc. British Machine Vision Conference, London, UK*, Sept. 2009.

[42] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, June 2004.

[43] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, June 2007.

[44] P. Sturm and B. Triggs, "A factorization-based algorithm for multi-image projective structure and motion," *Proc. European Conf. Computer Vision, Cambridge, UK*, vol. 1, pp. 709–720, Apr. 1996.

[45] C. Fruh and A. Zakhor, "An automated method for large-scale, ground-based city model acquisition," *Int'l Journal Computer Vision*, vol. 60, no. 1, pp. 5–24, Oct 2004.

[46] C. Frueh and A. Zakhor, "Constructing 3D city models by merging ground-based and airborne views," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2003.

[47] A. Akbarzadeh, "Towards urban 3D reconstruction from video," in *Int'l Symposium on 3D Data, Processing, Visualization and Transmission*, 2006.

[48] R. Carceroni, A. Kumar, and K. Daniilidis, "Structure from motion with known camera positions," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition, Washington DC*, vol. 1, June 2006.

[49] G. Qian, Q. Zheng, and R. Chellappa, "Reduction of inherent ambiguities in structure from motion problem using inertial data," in *Proc. Int'l Conf. Image Processing, Vancouver, Canada*, vol. 1, Sept. 2000.

[50] M. Irani, P. Anandan, and M. Cohen, "Direct recovery of planar-parallax from multiple frames," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1528–1534, 2002.

[51] C. Rother and S. Carlsson, "Linear multi view reconstruction and camera recovery using a reference plane," *Int'l Journal Computer Vision*, vol. 49, no. 3, pp. 117–141, Sept. 2002.

[52] R. Kaucic, R. Hartley, and N. Dano, "Plane-based projective reconstruction," in *Proc. Int'l Conf. Computer Vision, Vancouver, Cananda*, vol. 1, July 2001.

[53] A. Bartoli, "The geometry of dynamic scenes - on coplanar and convergent linear motions embedded in 3d static scenes," *Computer Vision and Image Understanding*, vol. 98, no. 2, pp. 223–238, May 2005.

[54] A. Fitzgibbon and A. Zisserman, "Multibody structure and motion: 3-d reconstruction of independently moving objects," in *Proc. European Conf. Computer Vision*, vol. 98, no. 2, May 2000, pp. 223–238.

[55] J. Li and R. Chellappa, "Structure from planar motion," *IEEE Trans. Image Processing*, vol. 12, no. 1, pp. 234–778, November 2006.

[56] S. Mahamud, M. Hebert, Y. Omori, and J. Ponce, "Provably-convergent iterative methods for projective structure from motion," *Proc. IEEE Conf. Computer Vision and Pattern Recognition, Honolulu, Hawaii*, vol. 1, Dec. 2001.

[57] J. Oliensis and R. Hartley, "Iterative extensions of the sturm/triggs algorithm: Convergence and nonconvergence," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, pp. 2217–2233, Dec. 2007.

[58] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," in *Proc. European Conf. on Computer Vision*, 1992, pp. 237–252.

[59] M. Irani and P. Anandan, "Robust multi-sensor image alignment," in *Proc. IEEE Int'l Conf. on Computer Vision*, January 1998, pp. 959–966.

[60] A. Zomet, A. Levin, S. Peleg, and Y. Weiss, "Seamless image stitching by minimizing false edges," *IEEE Trans. Image Processing*, vol. 15, no. 4, pp. 969–977, Apr 2006.

[61] J. Pluim, J. Maintz, and M. Viergever, "Image registration by maximization of combined mutual information and gradient information," *IEEE Trans. Medical Imaging*, vol. 19, no. 8, pp. 809–814, Aug 2000.

[62] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[63] M. Ramachandran and R. Chellappa, "Gradient and feature-based video stabilization," *Submitted to IEEE Trans. Image Processing*, 2010.

[64] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1778 –1792, Nov. 2005.

[65] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.

[66] Y. Boykov, O. Veksler, and R. Zabih, "Efficient approximate energy minimization via graph cuts," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1222–1239, Nov. 2001.

[67] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, Feb. 2004.

[68] C. Shekhar, "Semi-automatic video-to-site registration for aerial monitoring," in *Proc. IEEE Int'l Conf. on Pattern Recognition*, January 1998, pp. 959–966.

[69] M. Ramachandran, A. Veeraraghavan, and R. Chellappa, "Fast bilinear SfM with side information," in *Proc. IEEE Int'l Conf. Computer Vision, Rio de Janeiro, Brazil*, Oct. 2007.

[70] ——, “A fast bilinear structure from motion algorithm using a video sequence and inertial sensors,” *Under revision in IEEE Trans. Pattern Analysis and Machine Intelligence*, 2010.

[71] E. Malis and R. Cipolla, “Camera calibration from unknown planar structures enforcing the multi view constraints between collineations,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1268–1272, Sept. 2002.

[72] B. Triggs, P. F. Mclauchlan, R. I. Hartley, and A. W. Fitzgibbon, *Bundle Adjustment – A Modern Synthesis*, January 2000, vol. 1883. [Online]. Available: http://www.metapress.com/link.asp?id=PLVCRQ5BX753A2TN

[73] C. Yuan and G. Medioni, “3D reconstruction of background and objects moving on ground plane viewed from a moving camera,” in *Proc. Int’l Conf. Computer Vision and Pattern Recognition*, 2006.

[74] A. Buchanan and A. Fitzgibbon, “Damped Newton algorithms for matrix factorization with missing data,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition, San Diego, CA*, vol. 2, June 2005.

[75] M. Lourakis, “Levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++,” http://www.ics.forth.gr/∼lourakis/levmar/, Jul. 2004.

[76] W. W. Hager and H. Zhang, “CG DESCENT: A conjugate gradient method with guaranteed descent,” *ACM Trans. Math. Software*, vol. 32, pp. 113–137, Mar. 2006.

[77] D. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Addison-Wesley, 1984.

[78] C.-P. Lu, G. Hager, and E. Mjolsness, “Fast and globally convergent pose estimation from video images,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 610–622, Jun 2000.