ROBUST EXTRACTION OF SPARSE 3D POINTS FROM IMAGE SEQUENCES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ELİF VURAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2008

Approval of the Thesis

ROBUST EXTRACTION OF SPARSE 3D POINTS FROM IMAGE SEQUENCES

Submitted by **ELİF VURAL** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İsmet Erkmen
Head of Department, **Electrical and Electronics Engineering** _____

Assoc. Prof. Dr. A. Aydın Alatan
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering, METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering, METU _____

M.S. Salim Sırtkaya
ASELSAN A.Ş. _____

Date: 03.09.2008

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Lastname : Elif Vural

Signature :

ABSTRACT

ROBUST EXTRACTION OF SPARSE 3D POINTS FROM IMAGE SEQUENCES

Vural, Elif
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. A. Aydın Alatan

September 2008, 144 pages

In this thesis, the extraction of sparse 3D points from calibrated image sequences is studied. The presented method for sparse 3D reconstruction is examined in two steps, where the first part addresses the problem of two-view reconstruction, and the second part is the extension of the two-view reconstruction algorithm to include multiple views. The examined two-view reconstruction method consists of some basic building blocks, such as feature detection and matching, epipolar geometry estimation, and the reconstruction of cameras and scene structure. Feature detection and matching is achieved by Scale Invariant Feature Transform (SIFT) method. For the estimation of epipolar geometry, the 7-point and 8-point algorithms are examined for Fundamental matrix (F-matrix) computation, while RANSAC and PROSAC are utilized for the robustness and accuracy for model estimation. In the final stage of two-view reconstruction, the camera projection matrices are computed from the F-matrix, and the locations of 3D scene points are estimated by triangulation; hence, determining the scene structure and cameras up to a projective transformation. The extension of the two-view reconstruction to multiple views is achieved by estimating the camera projection matrix of each

additional view from the already reconstructed matches, and then adding new points to the scene structure by triangulating the unreconstructed matches. Finally, the reconstruction is upgraded from projective to metric by a rectifying homography computed from the camera calibration information. In order to obtain a refined reconstruction, two different methods are suggested for the removal of erroneous points from the scene structure. In addition to the examination of the solution to the reconstruction problem, experiments have been conducted that compare the performances of competing algorithms used in various stages of reconstruction. In connection with sparse reconstruction, a rate-distortion efficient piecewise planar scene representation algorithm that generates mesh models of scenes from reconstructed point clouds is examined, and its performance is evaluated through experiments.

Keywords: Sparse 3D scene reconstruction, structure-from-motion.

ÖZ

SEYREK 3B NOKTALARIN GÖRÜNTÜ DİZİLERİNDEN DAYANIKLI BİÇİMDE ÇIKARIMI

Vural, Elif
Yüksek Lisans, Elektrik-Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. A. Aydın Alatan

Eylül 2008, 144 sayfa

Bu tezde kalibre edilmiş görüntü dizilerinden 3B noktaların çıkarımı incelenmiştir. 3B sahne geriçatımı için önerilen yöntem, ilki iki görüntüden geriçatım problemi, ikincisi ise bunun çok sayıda görüntüye uyarlanması olmak üzere iki aşamada ele alınmıştır. İncelenen iki görüntüden geriçatım yöntemi; öznitelik tespiti ve eşleme, eşkutupsal geometri çıkarımı ve kamera ve sahne yapısının geriçatımı gibi temel bileşenlerden oluşmaktadır. Öznitelik tespiti ve eşleme, SIFT yöntemiyle gerçekleştirilmiştir. Eşkutupsal geometri çıkarımı ile ilgili olarak, temel matris hesaplanması amacıyla 7-nokta ve 8-nokta algoritmaları incelenmiş, ayrıca dayanıklı ve doğru model çıkarımı için RANSAC ve PROSAC kullanılmıştır. İki görüntüden geriçatım probleminin son basamağında ise, temel matristen kamera izdüşüm matrisleri hesaplanır ve üçgenleme yöntemiyle 3B sahne noktalarının yerleri saptanır; böylece sahne yapısı ve kameralar izdüşümsel bir dönüşümün belirsizlik sınırları dahilinde tespit edilmiş olur. İki görüntüden geriçatımın çoklu görüntülere uyarlanması, her yeni görüntü için, kamera izdüşüm matrisinin geriçatımı hâlihazırda gerçekleştirilmiş nokta çiftlerinden hesaplanması, ve daha sonra sahne yapısına henüz geriçatılmamış nokta çiftlerinin

üçgenlenerek eklenmesi ile gerçekleştirilir. Son olarak, kamera kalibrasyon bilgisinden elde edilen doğrultucu bir homografi ile geriçatımın izdüşümselden metriğe çevrimi sağlanır. Temiz bir geriçatım elde edebilmek amacıyla, yanlış noktaların sahneden çıkarılmasına yönelik iki ayrı yöntem önerilmiştir. Geriçatım probleminin çözümüne yönelik incelemelerin yanı sıra, geriçatımın çeşitli basamaklarında kullanılan rakip algoritmaların performanslarını karşılaştıran deneyler de yapılmıştır. Seyrek geriçatım problemiyle bağlantılı olacak şekilde, sahnenin geriçatılmış nokta bulutlarından örgü modelini oluşturan ve hız-bozunum bağlamında verimli gösterimini hedefleyen bir algoritma incelenmiş, ayrıca algoritmanın performansı deneylerle irdelenmiştir.

Anahtar kelimeler: Seyrek 3B sahne geriçatımı, hareketten-yapı.

To the memory of my cousin Tuna Yaşar

ACKNOWLEDGEMENTS

Firstly, I would like to thank my thesis supervisor, Assoc. Prof. Dr. A. Aydın Alatan, for his guidance, understanding and consideration. His broad vision and motivation facilitated the progress of my studies immensely, and I regard it as a privilege to work with him.

Special thanks to Evren İmre, not only for his contribution to this thesis, but also for his patience, friendship and for many discussions we had on both technical and non-technical matters. I learnt a great deal from him.

Thanks to each one of my officemates. It is their companion that made my time in MMRG so enjoyable and valuable.

Finally, I would like to thank my family and Eren Şems Halıcı, for their love and support.

This work is partially funded by EC 6th Framework Program IST 3DTV NoE and TÜBİTAK under Career Project 104E022.

TABLE OF CONTENTS

ABSTRACT iv
ÖZ vi
ACKNOWLEDGEMENTS ix
TABLE OF CONTENTS x
CHAPTERS
1 INTRODUCTION 1
1.1 Literature Review 2
1.2 Overview of the Thesis 4
1.3 Outline of the Thesis 5
2 3D SPARSE RECONSTRUCTION OF A SCENE FROM TWO VIEWS 7
2.1 An Overview of Sparse 3D Reconstruction 7
2.2 Feature Detection and Matching 10
2.2.1 Scale Invariant Feature Transform 11
2.2.2 Harris Corner Detector 25
2.3 Epipolar Geometry Estimation 28
2.3.1 An Overview of Epipolar Geometry 29
2.3.2 Computation of the Fundamental Matrix from Correspondences 33
2.3.2.1 Seven-Point Algorithm 35
2.3.2.2 Eight-Point Algorithm 35
2.3.2.3 Normalization of Correspondence Coordinates 37
2.3.3 Sampling Strategies in Fundamental Matrix Estimation 40
2.3.3.1 Random Sample Consensus (RANSAC) 41
2.3.3.2 Progressive Sample Consensus (PROSAC) 46
2.4 Reconstruction of Cameras and Structure 53
2.4.1 An Overview of Projective Geometry 54
2.4.2 Projective Reconstruction of the Scene 61
2.4.2.1 Determination of Projection Matrices 61

2.4.2.2 Triangulation 62
2.4.3 Upgrading the Projective Reconstruction to Metric 69
2.4.4 Experimental Results 72
2.5 Experimental Comparison of Competing Algorithms Used in Sparse Reconstruction 76
2.5.1 Experimental Comparison of SIFT, Harris Corner Detector and KLT Tracker 76
2.5.2 Experimental Comparison of RANSAC and PROSAC 81
2.5.3 Experimental Comparison of 7-point and 8-point Algorithms 87
3 3D SPARSE RECONSTRUCTION OF A SCENE FROM MULTIPLE VIEWS 93
3.1 Multiple View Reconstruction 94
3.1.1 Overview of the Multiple View Reconstruction Algorithm 94
3.1.2 Estimation of the Projection Matrix from 3D Points and Their 2D Projections 97
3.1.3 Experimental Results 100
3.2 Removal of Outliers from the Reconstruction 105
3.2.1 Outlier Elimination via Optimization 106
3.2.2 Intensity Correlation Based Outlier Elimination 110
3.3 Sparse Bundle Adjustment of the Reconstruction Parameters 113
4 MESH-BASED SCENE REPRESENTATION WITH APPLICATIONS TO VIDEO ENCODING 118
4.1 Introduction 118
4.2 Rate-Distortion Efficient Scene Representation 119
4.2.1 Algorithm 119
4.2.2 Experimental Results 125
5 CONCLUSIONS 134
APPENDIX A 138
APPENDIX B 139
REFERENCES 141

CHAPTER 1

INTRODUCTION

In the last few decades, there has been growing interest in the extraction, transmission and display of 3D visual data. Although 2D imaging technologies still prevail in the current market, the effort made towards the deployment of 3D information in the related industrial fields is noteworthy. For example, 3D movies are becoming increasingly popular now, and there is demand for the modeling of real 3D scenes in virtual environments, to be used in many applications such as online databases, computer games or educational simulators.

One of the popular research topics is the conversion of the already available 2D video content to 3D, in order to enable the usage of ordinary video data in several 3D systems, such as free viewpoint television (FTV) and three dimensional television (3DTV). While a considerable amount of progress has been made towards the extraction of 3D structure from 2D image data, the problem of processing an ordinary 2D video to extract the 3D information in the captured scenes in a fully automatic manner is still a challenge.

A visually meaningful representation of a scene requires the recovery of the 3D information on all scene surfaces, or the extraction of the dense depth map of the scene. While *dense 3D reconstruction* is a research field having this objective, *sparse 3D reconstruction* is another sub-field of the scene reconstruction problem,

where the output provided by a sparse 3D reconstruction algorithm is not a 3D surface but a 3D point cloud, which is infeasible to use directly in practical applications such as FTV or 3DTV. However, the importance of sparse 3D reconstruction lies in the fact that it is an effective tool for the extraction of camera poses and projection matrices from an already recorded uncontrollable data, which can provide useful information for the next step, namely, dense surface reconstruction.

In this thesis, the problem of reconstructing a static 3D scene sparsely from an image sequence captured with a calibrated camera is studied. In the presented sparse 3D reconstruction scheme, the internal calibration information for the viewing cameras is assumed to be known, where the goal is to retrieve the 3D locations of some sparsely selected scene points and the camera poses, i.e. the *camera projection matrices* of the viewing cameras, given an image sequence viewing the scene.

1.1 Literature Review

The problem of the extraction of 3D information from 2D visual data has been drawing quite much interest in the recent years. A considerable amount of work has been done in the literature related to this field, some of which deal with the problem of constructing a complete system that retrieves 3D information from 2D data, whereas some other works concentrate on separate stages of reconstruction.

In [24], Pollefeys et al. present a successful example of a complete system that is capable of building the 3D model of a scene captured with an uncalibrated hand-held camera. The system reconstructs the sparse scene structure and the camera poses from tracked or matched features and converts the projective reconstruction to metric through self-calibration. The dense scene structure is also estimated by

using stereo-matching tools, and the proposed system achieves highly satisfactory reconstruction results.

A similar work is presented by İmre et al. [25], which reconstructs 3D scenes from broadcast videos. The feature detection, sparse reconstruction and self-calibration steps are again included similar to the work of Pollefeys et al. [24], but the dense depth estimation is achieved with MRF-based techniques. A prioritization scheme is also proposed for registering the images to the reconstruction.

The initial step for sparse reconstruction algorithms is detection of features. A combined corner and edge detection algorithm was proposed by Harris and Stephens in 1988 [3], which is now generally referred to as Harris corner detector, and widely used for feature detection in computer vision. A recent novelty in feature detection is Scale Invariant Feature Transform (SIFT) [4], proposed by Lowe, whose most prominent improvements on Harris corner detector is robustness against scale change and rotation. A convenient and accurate feature matching mechanism is also proposed with this algorithm.

Another major stage of sparse reconstruction is the estimation of epipolar geometry from feature correspondences. 7-point and 8-point algorithms [1],[10] are known methods for the computation of the fundamental matrix, which models the epipolar geometry, from correspondences. 8-point algorithm has been initially proposed by Longuet-Higgins [10], and in [11] Hartley has shown that a simple normalization of correspondence coordinates brings significant improvements on the performance of the algorithm.

In practical reconstruction problems, the estimation of epipolar geometry from correspondences also requires the utilization of an algorithm for efficient sampling of correspondences for the computation of the fundamental matrix. Random Sampling Consensus (RANSAC) [12], proposed by Fischler and Bolles

in 1981, is a well-known algorithm used for this purpose, which is designed for computing the correct model such that the solution is unaffected by gross errors in the input data. Many other algorithms resembling RANSAC have been developed in the following years, such as PROSAC [13], DEGENSAC [14] and MLESAC [15], each of which provides different improvements on the performance of RANSAC.

1.2 Overview of the Thesis

In this thesis, a solution is proposed to the problem of sparse reconstruction of a scene from an image sequence captured with a camera of known internal parameters. The reconstruction is achieved by first performing a two-view reconstruction of the scene, utilizing the first two images in the sequence, then extending the reconstruction to multiple views by registering the remaining frames.

The two-view reconstruction procedure is composed of the following blocks: Firstly features are detected on both images and matched to obtain correspondences. Then the Fundamental matrix (F-matrix), which is the algebraic representation of the epipolar geometry between the two views, is computed from the correspondence pairs. In order to reconstruct the scene structure and the cameras, projective camera matrices are computed from the F-matrix, and then the matches which are consistent with the model are triangulated to find the locations of the corresponding 3D points up to a projective transformation.

In order to upgrade the two-view reconstruction to include multiple views, features are detected in each additional image and matched to the features in the previous image. The projective camera matrices are computed from the already reconstructed matches, and then new points are added to the reconstruction by triangulating the matches that have not been reconstructed earlier. Finally, the

projective scene reconstruction is transformed to metric by using the camera calibration information.

The sparse reconstruction of the scene is refined by removing the erroneous points from the scene with two different novel methods. Finally, a sparse bundle adjustment algorithm is utilized to optimize the camera poses and the reconstructed point locations.

In addition to studying the sparse reconstruction problem, the rate-distortion efficient scene representation algorithm proposed by İmre [28] is examined, which is a method of upgrading the sparse reconstruction to a mesh-based dense reconstruction in a rate-distortion efficient manner. The performance of the examined algorithm is also investigated via experiments.

1.3 Outline of the Thesis

The organization of the thesis is as follows: In Chapter 2, the two-view reconstruction problem is discussed. Each of the building blocks of two-view reconstruction, namely, feature detection and matching, epipolar geometry estimation and reconstruction of the cameras and the structure, are examined in detail. The chapter ends with the results of some experiments comparing the performances of opponent algorithms used during reconstruction.

In Chapter 3, multiple-view reconstruction is explained and reconstruction results for several image sequences are presented. The chapter continues with the discussion of two different methods for the elimination of outliers from the reconstruction. Finally, the optimization of the reconstruction parameters through a sparse bundle adjustment algorithm is explained.

Chapter 4 is devoted to the examination of the rate-distortion efficient piecewise planar scene representation algorithm proposed by İmre [28]. An overview of the algorithm is presented, as well as the results of some experiments on the performance of the algorithm.

The thesis is concluded with the final discussions and comments in Chapter 5.

CHAPTER 2

3D SPARSE RECONSTRUCTION OF A SCENE FROM TWO VIEWS

In this chapter, the problem of sparse 3D metric reconstruction of a static scene from two views of the scene is studied. The purpose of the presented algorithm is to determine the 3D locations of some sparsely selected points in the scene, given two calibrated views of the scene, meaning that the internal parameters of the viewing cameras are available. In addition to obtaining a 3D point cloud representing the sparsely selected scene points, the camera poses are estimated as well.

The organization of this chapter is as follows: In Section 2.1, an outline of the studied sparse 3D reconstruction method is presented and the main blocks of the algorithm are briefly examined. In Sections 2.2, 2.3 and 2.4, these building blocks are explained in detail, whereas the results of several experiments comparing the performances of competing algorithms used in sparse 3D reconstruction are discussed in Section 2.5.

2.1 An Overview of Sparse 3D Reconstruction

The studied method for sparse 3D reconstruction, which takes two images viewing the same scene as input, and constructs a 3D point cloud representing the scene as output, consists of three main blocks:

- Feature detection and matching
- Epipolar geometry estimation
- Reconstruction of the cameras and the structure.

The overall structure of the reconstruction algorithm is illustrated in Figure 2.1.

The first stage of reconstruction is feature extraction and matching. The purpose of sparse reconstruction is to obtain the 3D information only at some sparsely selected scene points. Therefore, a natural initial step for sparse 3D reconstruction is the determination of some salient interest points in both views, and matching these selected points to obtain correspondence pairs, which are called feature detection and feature matching respectively. The problem of feature detection and matching is discussed in Section 2.2.

The next stage following the feature detection and matching is the estimation of the epipolar geometry relating the two views geometrically. The determination of epipolar geometry requires at least seven valid correspondence pairs (matches). Epipolar geometry is modeled by the F-matrix.

The final stage of reconstruction is the estimation of the 3D scene structure and the camera poses. The F-matrix relating the two image planes gives clues about the camera poses, and it is possible to infer camera poses by utilizing the estimated epipolar geometry together with the internal camera calibration information. Once the camera poses, i.e. the projection matrices corresponding to the two cameras, are determined, it is possible to estimate the 3D locations of scene points, each of which corresponds to a match, via triangulation.

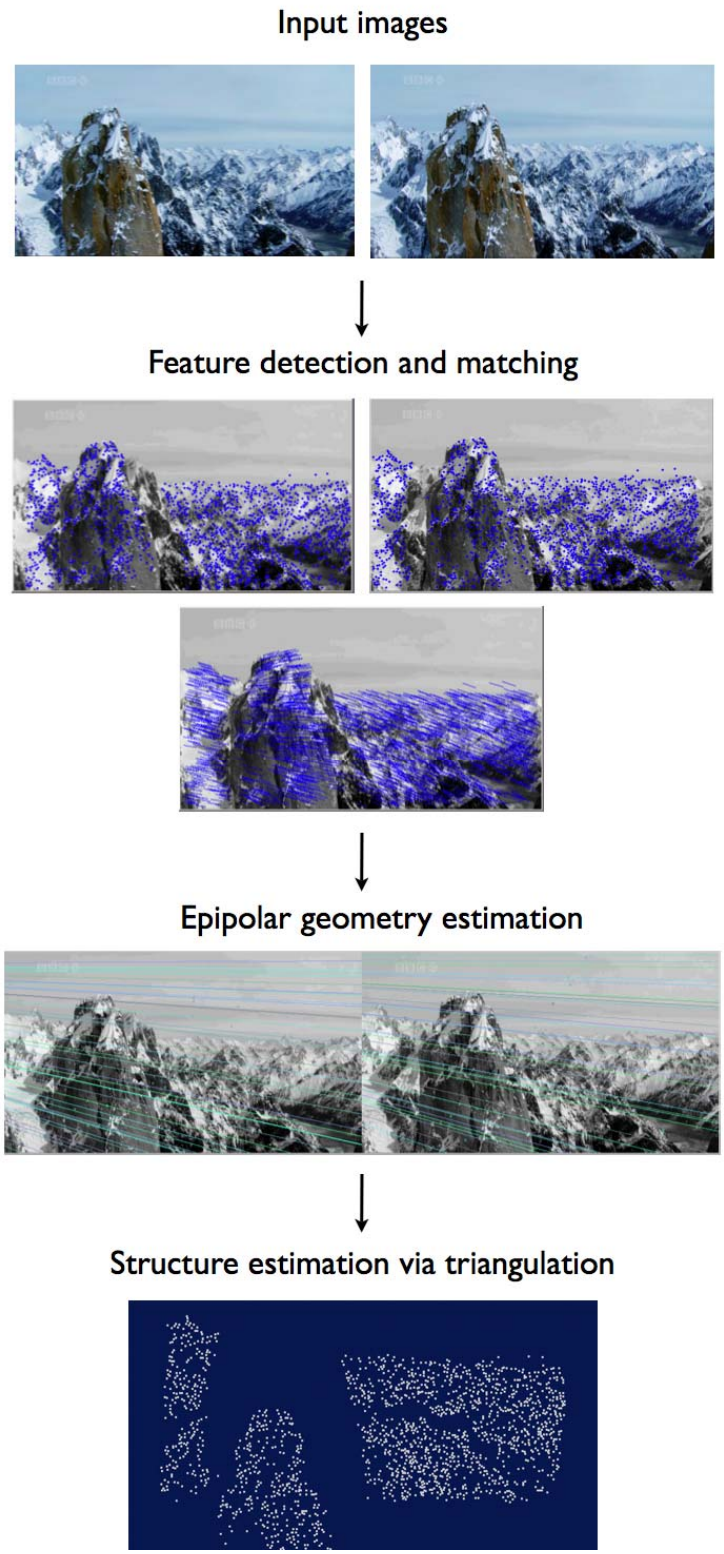


Figure 2.1: Main steps of sparse 3D reconstruction problem

2.2 Feature Detection and Matching

In this work, the estimation of epipolar geometry, which geometrically relates the image planes belonging to two views, is achieved by utilization of correspondence pairs. A correspondence pair, i.e. a match, is a pair of 2D points which are the projections of the same 3D scene point, where one of the points lies on the first image plane, and the other on the second image plane. The illustration of a correspondence pair is given in Figure 2.2.

The formation of such matches requires, firstly the determination of some specific, salient points on both images, which have some special characteristics distinguishing themselves from other image points. The determination of points with such properties is denoted as feature detection. Feature detection is followed by feature matching, which is the problem of forming a set of correspondence pairs, given two sets of features each belonging to one of the images.

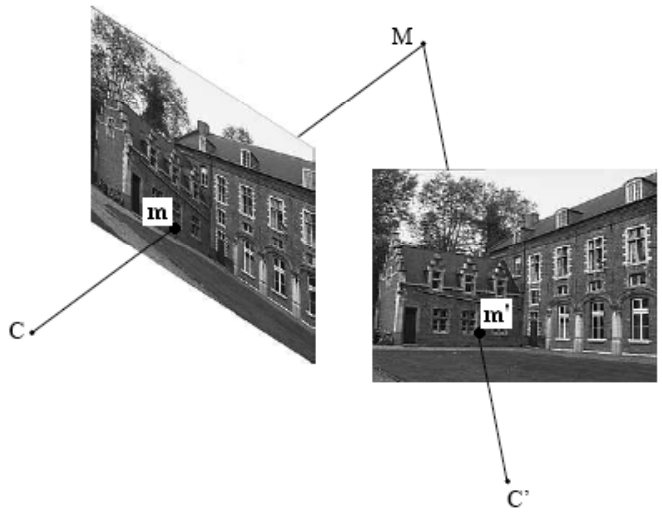


Figure 2.2: Illustration of a pair of matches. M denotes a 3D point in the scene, C and C' are the camera centers of the two viewing cameras, and (m,m') is the correspondence pair belonging to M [2].

In the literature several methods have been developed for feature detection. In 1988, Harris and Stephens proposed a corner detection algorithm [3] based on defining a *cornerness measure* as a function of image intensity gradients and determining the corners accordingly. Since then, Harris corner detector has been a popular and widely used tool for feature detection purposes.

An important development in feature detection has been the introduction of Scale Invariant Feature Transform (SIFT) to the literature, which was proposed by Lowe in 2004 [4]. The most celebrated improvements provided by SIFT are invariance to scale and rotation, and accuracy in feature point localization and matching. However, the computational complexity of SIFT usually discourages its real-time utilization and restricts its usage to off-line applications.

In this thesis, feature detection is achieved with Scale Invariant Feature Transform. SIFT is explained in Section 2.2.1, whereas a brief overview to Harris corner detector is given in Section 2.2.2. In order to assess the performance of SIFT, some experiments have been conducted comparing SIFT to Harris corner detector and KLT tracker [20], the results of which are presented in Section 2.5.1.

2.2.1 Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) is a method proposed by David Lowe [4], for the extraction of distinctive features from images. One great motivation behind the development of SIFT is the goal of assuring scale invariance in feature detection. In order to perform feature detection with invariance to scale change, the image is searched for stable features at all possible scales.

The approach adopted in SIFT is based on forming the scale space of an image by convolving the image with a sequence of 2D Gaussian functions at different scales, and computing difference-of-Gaussian-convolved (DoG-convolved)

images from the differences of Gaussian-convolved images at adjacent scales. The features, called *keypoints*, are then detected by searching the scale space extrema of the DoG-convolved images.

Along with its amenability to computation, the reason behind the selection of this particular approach for feature detection has some mathematical aspects as well. Before the explanation of the theoretical content of SIFT, it may be helpful to review some main concepts from scale-space theory [6].

Given a continuous signal $f:\mathbb{R}^{\mathrm{N}}\to\mathbb{R}$, the scale-space representation of *f*, $L:\mathbb{R}^{\mathrm{N}}\times\mathbb{R}^{+}\to\mathbb{R}$ is defined as the solution to following differential equation known as the *heat diffusion equation* [6],

$$\partial_t L=\frac{1}{2}\nabla^2 L=\frac{1}{2}\sum_{i=1}^{\mathrm{N}}\partial_{x_i x_i}L \tag{2.1}$$

with initial condition $L(\,.\,;0)=f(.)$. The family of solutions to the diffusion equation can equivalently be defined by convolution with Gaussian kernels of various width *t,*

$$L(.;t)=g(.;t)*f(.), \tag{2.2}$$

where $g:\mathbb{R}^{\mathrm{N}}\times\mathbb{R}^{+}\to\mathbb{R}$ is given by

$$g(\mathbf{x};t)=\frac{1}{(2\pi t)^{\mathrm{N}/2}}e^{-(x_1^2+\ldots+x_{\mathrm{N}}^2)/(2t)}, \tag{2.3}$$

and $\mathbf{x}=(x_1,\ldots,x_{\mathrm{N}})$. There are several mathematical results showing that within the class of linear transformations, Gaussian kernel is the unique kernel for generating

a scale-space satisfying some basic conditions such as linearity, shift invariance, etc. [6].

In 1983, Witkin showed that the detection of scale- invariant features in an image is achievable by using the scale-space of the image [5]. A property of scale-space is that the amplitude of the scale-space representation of a function generally decreases with scale [6]. In consistency with this observation, Lindeberg studied the scale-normalized Laplacian-of-Gaussian function $\sigma^2\nabla^2 G$ to show that the normalization of the Laplacian with the factor σ^2 is required for scale invariance [7],[4].

It has been shown by experiments that the extrema of $\sigma^2\nabla^2 G$ produce the most stable features compared to a range of other image functions [8],[4]. A critical property of the difference-of-Gaussian (DoG) function is that it provides a close approximation to the scale-normalized Laplacian-of-Gaussian function $\sigma^2\nabla^2 G$ [4]:

Expressing the diffusion equation in terms of σ instead of *t*, the 2D Gaussian function is defined as

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}, \tag{2.4}$$

and the following relations can be derived:

$$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{2.5}$$

Equation (2.5) yields the approximation:

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G \,. \tag{2.6}$$

Equation (2.6) shows that, if a constant factor of k is used between the scales of adjacent difference-of-Gaussian functions, i.e. the two nearby scales are always in the form of σ and $k\sigma$, then the difference-of-Gaussian function implicitly incorporates the scale normalization σ^2 required for scale-invariance. Since the factor (k-1) is constant, it does not affect the extrema location.

Detection and Localization of Features

In order to detect features in an image, the first step is to obtain the scale-space $L(x,y,\sigma)$ of the input image $I(x,y)$ by convolving the input image with a variable-scale Gaussian image:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.7}$$

The DoG-convolved images $D(x,y,\sigma)$ can be computed from the difference of two scale-space images with adjacent scales differing by a factor of k:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ D(x, y, \sigma) &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{2.8}$$

For the construction of $D(x,y,\sigma)$, first the input image $I(x,y)$ is convolved with a sequence of Gaussian images $G(x,y,\sigma)$ to obtain the scale-space images $L(x,y,\sigma)$. The sequence of Gaussian images is organized in groups, called octaves, where in an octave of Gaussian images, the scale of each Gaussian is always k times the scale of the previous one. Each octave consists of s intervals, where s is an integer, and s and k are related by $k = 2^{1/s}$. For the ease of local extrema detection, s+3 images are produced for each octave. The DoG-convolved images

$D(x,y,\sigma)$ are obtained by subtracting the scale-space images $L(x,y,\sigma)$ with adjacent scales.

When a whole octave of images is processed in this way to obtain DoG-convolved images, the algorithm passes to the next octave. When passing from an octave to the next one, the last scale-space image of the octave is downsampled by two to obtain the first scale-space image of the next octave. The described process is depicted in Figure 2.3.

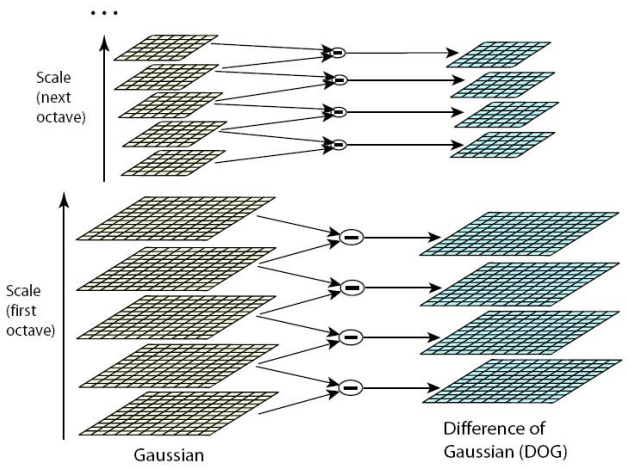


Figure 2.3: In each octave, the input image is convolved with a sequence of Gaussian images where adjacent images have scales differing by a multiplicative factor of *k*. After each octave, the last (top) image in the octave is downsampled by two to obtain the first (bottom) image of the next octave. After the scale-space of an image is obtained in this way by convolving with a sequence of Gaussian images, the DoG-convolved image sequence is obtained by subtracting the adjacent scale-space images in each octave [4].

Once the DoG-convolved images are constructed, the detection of features is achieved by searching the DoG-convolved images for local extrema. If the image intensity value at a pixel is the maximum or minimum among all of its 26 neighbours, i.e. 8 neighbours on the same image and 9 neighbours on each of the two adjacent images, then that pixel is marked as a feature (keypoint) candidate, which is illustrated in Figure 2.4.

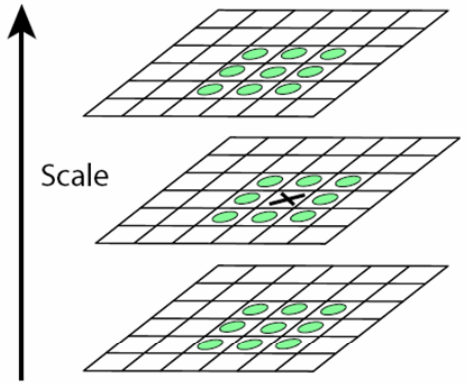
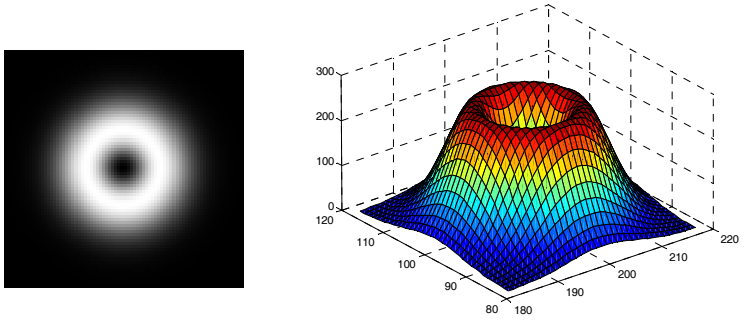


Figure 2.4: Detection of features by searching the DoG-convolved images: If the pixel marked with X has an intensity value greater than or smaller than all its 26 neighbours marked with green dots, then a feature candidate is detected at that pixel location [4].

The fact that SIFT detects features by searching for the extrema in the DoG-convolved images, practically means that SIFT is in favour of image regions similar to a DoG (difference-of-Gaussian) function, whose shape resembles a ring or a Mexican hat. The intensity image and graphical representations of a 2D difference-of-Gaussian function are given in Figure 2.5.



(a) **(b)**

Figure 2.5: (a) The intensity image representation of a 2D DoG function **(b)** The graphical representation of a 2D DoG function

Since the scale-space of the input image is generated at various scales, SIFT is capable of detecting features at a wide range of scales. In order to clarify the scale-invariance property of SIFT better, some SIFT features that are obtained on a sample image are shown in Figure 2.6. The image intensity surfaces are also plotted in regions around the feature locations. It is noticeable from the plots that the image intensities around features resemble difference-of-Gaussian functions (or equivalently reversed DoG functions), and the scales at which features are detected show variation as well.

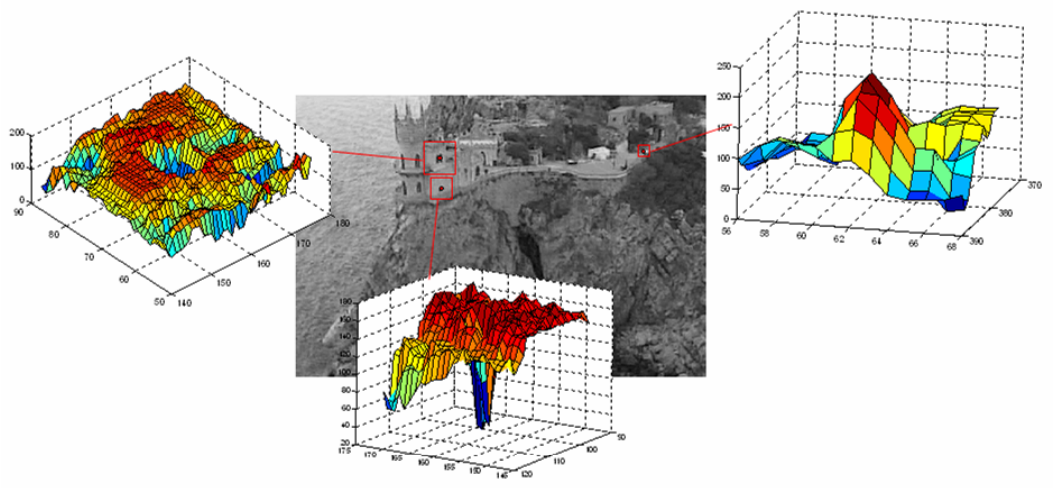


Figure 2.6: Some SIFT features detected on a typical image, and the image intensity plots around feature locations

Once feature candidates are determined, the next step is the accurate localization of features (keypoints). The approach of detecting keypoints by comparing pixels to their 26 nearest neighbours gives the keypoint locations only with pixel accuracy. The subpixel localization of a keypoint is realized by fitting a 3D quadratic function to the image intensity function around the keypoint at the corresponding DoG-convolved image, and determining the local extremum of the modeling function. Fitting function is obtained from the Taylor series expansion of the image intensity function around the origin, where the origin is assumed to be the center of the pixel the original keypoint resides at. The 3D quadratic fitting

function $\hat{D}(\mathbf{X})$ to the intensity function of the DoG-convolved image $D(\mathbf{X}) = D(x,y,\sigma)$ is

$$\hat{D}(\mathbf{X}) = D(\mathbf{X}_0) + (\nabla_{\mathbf{X}_0} D)^{\mathrm{T}} \mathbf{X} + \frac{1}{2} \mathbf{X}^{\mathrm{T}} (\mathbf{H}_{\mathbf{X}_0} D) \mathbf{X}, \tag{2.9}$$

where $\mathbf{X_0} = (x_0, y_0, \sigma_0)$ is the pixel-accurate keypoint location taken as the origin, $\mathbf{X}=(x,y,\sigma)$ is the offset from the origin, $\nabla_{\mathbf{X}_0} D$ is the gradient of the function $D(\mathbf{X})$ at $\mathbf{X_0}$, and $\mathbf{H}_{\mathbf{X}_0} D$ is the Hessian matrix of the function $D(\mathbf{X})$ at $\mathbf{X_0}$. The subpixel-accurate location $\hat{\mathbf{X}}_\mathbf{0}$ of the keypoint is obtained by finding the local extremum of $\hat{D}(\mathbf{X})$, which is given by [4]

$$\hat{\mathbf{X}}_\mathbf{0} = -(\mathbf{H}_{\mathbf{X}_0} D)^{-1} (\nabla_{\mathbf{X}_0} D). \tag{2.10}$$

If the distance of $\hat{\mathbf{X}}_\mathbf{0}$ to $\mathbf{X_0}$ is larger than 0.5 pixels, then it is concluded that the pixel closest to the local extremum is not the one at location $\mathbf{X_0}$, but another one. In this case, the interpolation is repeated by taking the origin as the center of the closest pixel to $\hat{\mathbf{X}}_\mathbf{0}$ this time [4].

The classification of all keypoints obtained in this way as features may lead to the detection of unstable features. In order to prevent such situations, some mechanisms are required to eliminate unstable keypoints. The most straightforward check mechanism is to force the valid keypoints to have considerable image contrast. Therefore, if the image intensity function value at the local extremum, $D(\hat{\mathbf{X}}_\mathbf{0})$, is below some threshold, the keypoint detected at that location is discarded.

As well as thresholding the intensity function, another tool is also utilized in order to eliminate unstable keypoints. The DoG-convolved image has high intensity values in edge regions; however, since the localization of a keypoint on an edge would be highly susceptible to noise, keypoints associated with edges are considered to be unstable.

In order to check if a keypoint is associated with an edge, the principal curvatures of the DoG-convolved image at the keypoint location are examined, with the approach borrowed from the Harris corner detector [3]. On an edge, one of the principal curvatures has a high value and the other has a small value, whereas, on a stable keypoint the two principal curvatures are expected to be comparable. The principal curvatures of the function $D(x,y,\sigma)$ are proportional to the eigenvalues of the following matrix **H**, computed at the location and scale of the keypoint:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.11}$$

Since the ratio of the two principal curvatures is of interest rather than their individual values, it is enough to compute the ratio of the two eigenvalues. Denoting the larger eigenvalue of **H** by α, the smaller one by β, and their ratio by r, the following identities are obtained:

$$\begin{aligned} \mathrm{Tr}(\mathbf{H}) &= D_{xx} + D_{yy} = \alpha + \beta \\ \mathrm{Det}(\mathbf{H}) &= D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \end{aligned} \tag{2.12}$$

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{2.13}$$

The function $(r+1)^2 / r$ has a minimum if the two eigenvalues are equal, and is increasing with respect to r. Therefore, instead of computing r, $\mathrm{Tr}(\mathbf{H})^2 / \mathrm{Det}(\mathbf{H})$ is computed, and the keypoint is discarded if this value exceeds a threshold,

which means that the ratio of the principal curvatures is high and the keypoint is likely to be on an edge [4].

Descriptor Vector Assignment

The approach of SIFT for matching features is based on assigning a descriptor vector to each keypoint, which characterizes the keypoint, and then comparing the descriptor vectors of keypoints to match them. Therefore, the detection and localization of keypoints is followed by descriptor vector assignment.

The first step of descriptor assignment is orientation assignment. Orientations of keypoints are assigned with respect to gradient magnitude and direction. Once the keypoint orientations are determined, then the descriptor vector of each keypoint is constructed relative to its orientation, which is the key to the rotation invariance property of SIFT.

The orientation of a keypoint with coordinates (x_0,y_0,σ_0) is computed from the scale-space image $L(x,y,\sigma_0')$, where σ_0' is the closest available scale to σ_0. Computing the orientation from the image with the scale of the keypoint allows scale-invariance in feature matching. The magnitude $m(x,y)$ and direction $\theta(x,y)$ of the gradient of the scale-space image $L(x,y,\sigma_0')$ at location (x,y) are computed from pixel differences:

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2} \\ \theta(x,y) &= \tan^{-1}((L(x,y+1)-L(x,y-1))/(L(x+1,y)-L(x-1,y))) \end{aligned} \quad (2.14)$$

The gradient is computed in an image region around the keypoint, and then an orientation histogram is created by using these gradient computations. The 360° range of directions is quantized in 10° intervals to allow 36 bins in the histogram.

When adding a gradient measurement to the histogram, the measurement is weighted by the magnitude of the gradient and a weighting factor, where the weighting factor is determined by a circular Gaussian function with a standard deviation of 1.5 times the scale of the keypoint.

For the computation of the orientation of the keypoint, the highest peak in the orientation histogram is detected. In order to assign the orientation accurately, a parabola is fit to the histogram values at the bins corresponding to the peak and its two closest neighbours, and the peak position is interpolated, which determines the orientation of the keypoint. Occasionally, the orientation histogram might have more than one peak. If there are other peaks with values of at least 80% of the highest peak, then a separate keypoint is assigned for each of the peaks, which results in having several keypoints with the same location and scale but different orientations.

Once the orientation of the keypoint is determined, the descriptor vector is computed in the following way: The gradient magnitude and directions are determined on sample locations around the keypoint as explained. In order to assure rotation-invariance, gradient directions are taken relative to the orientation of the keypoint. Figure 2.7(a) illustrates the measurement of gradients on sample image locations.

Figure 2.7(a) shows the gradient measurements at sample locations on an 8x8 grid. This 8×8 grid is grouped into four 4×4 grids as shown in (a), and an orientation histogram is obtained from each of these four grids as shown in Figure 2.7(b). When forming the orientation histograms, the gradients are weighted by a circular Gaussian function with a standard deviation of half of the descriptor window width. The utilization of such a weighting function is useful in the sense that it lets the gradient measurements in image regions closer to the keypoint affect the descriptor more than more distant regions. Moreover, it also prevents

sudden changes in the descriptor due to the variations in keypoint localization. Each of the orientation histograms shown in Figure 2.7(b) has eight bins for gradient direction. The descriptor vector is obtained directly from these four orientation histograms by taking the value of each bin as an entry of the descriptor vector.

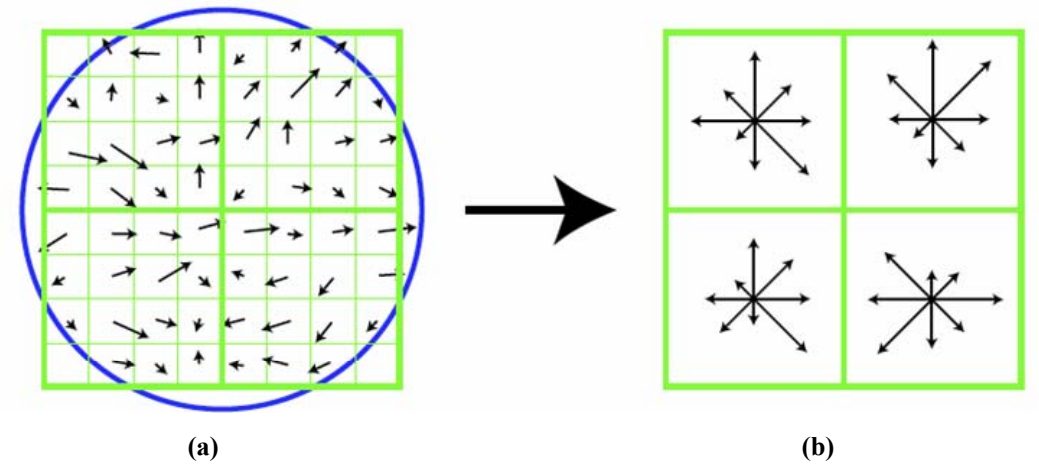


Figure 2.7: **(a)** Gradient measurements at sample locations on an 8×8 grid around the keypoint **(b)** 2×2 orientation histograms, each of which are obtained from a 4×4 grid [4]

In the process depicted in Figure 2.7, an 8×8 grid is used to sample image intensity gradients around the keypoint, and the descriptor vector is obtained from four (2×2) orientation histograms. However, in the SIFT implementation used in this thesis, image gradients are sampled on a 16×16 grid, and descriptor vectors are obtained from 16 (4×4) orientation histograms each with 8 bins, which results in a descriptor vector length of 16×8=128.

Algorithm Overview: Feature detection with SIFT

Input: An image
Output: Feature points (keypoints) detected on the image and their descriptor vectors

1. Obtain the scale-space of the input image by convolving with 2D Gaussian functions at different scales.

2. Subtract adjacent scale-space images to obtain the DoG-convolved image sequence.

3. Detect the extrema of the DoG-convolved image sequence and find the exact keypoint locations by using (2.9) and (2.10).

4. Discard the extrema with too low contrast or too high edge response.

5. Assign the orientations of the keypoints.

6. Compute orientation histograms in regions around the keypoints as depicted in Figure 2.7, and fill in the descriptor vectors using the orientation histograms.

Feature Matching

Feature detection stage is completed with the assignment of a descriptor vector to each keypoint. Once features are detected in both views of the scene, it is very simple to obtain correspondence pairs (matches) by the descriptor vectors.

Given a keypoint K_1 belonging to the first image, its match K_2 in the second image is determined by comparing the descriptor vectors of the keypoints in the second image to the descriptor vector of K_1. The keypoint in the second image having the descriptor vector with the smallest Euclidean distance in $\mathbb{R}^{128}$ to the

descriptor vector of K_1 is determined as its match K_2. It might sometimes happen that the descriptor vectors of the two strongest match candidates, i.e. the two closest vectors, may have comparable distances to the vector of K_1. This leads to an uncertainty in finding the correct match of K_1. In order to prevent such uncertain matches, the ratio d_1 / d_2 is calculated, where d_1 and d_2 are the distances of the closest vector and the next closest vector to the vector of K_1. The match is discarded if this ratio exceeds a predetermined threshold. Moreover, this ratio can also be regarded as a measure of the quality of the match, where a smaller ratio indicates a stronger match.

It was explained previously in Section 2.2.1 that there may be multiple keypoints at the same location but with different scales or orientations. In this case, feature matching is done regardless of duplicate keypoints, but when all correspondence pairs are determined, the list of correspondences is scanned to check if all duplicate keypoints from the first image are matched to keypoints having the same location in the second image. If there is any contradiction among the duplicates of a keypoint, then all matches corresponding to that keypoint are discarded. After the correspondence list is revised in this way to eliminate all contradicting duplicates, the list is reorganized such that all duplicates of a keypoint are represented by a single one; hence, only one correspondence pair is associated with a certain location in the first image.

Algorithm Overview: Feature matching with SIFT

Input: Feature points (keypoints) and their descriptor vectors belonging to two different images
Output: A list of correspondence pairs matching the features from the first and second images

1. a. For each keypoint in the first image, find the keypoint in the second image with the closest descriptor vector.

b. Add the match to the list of correspondences if the ratio of the distances of the two closest descriptor vectors is below some predefined threshold.

2. Discard the matches associated with duplicates of a keypoint if there is any contradiction between the matching locations of the duplicates.

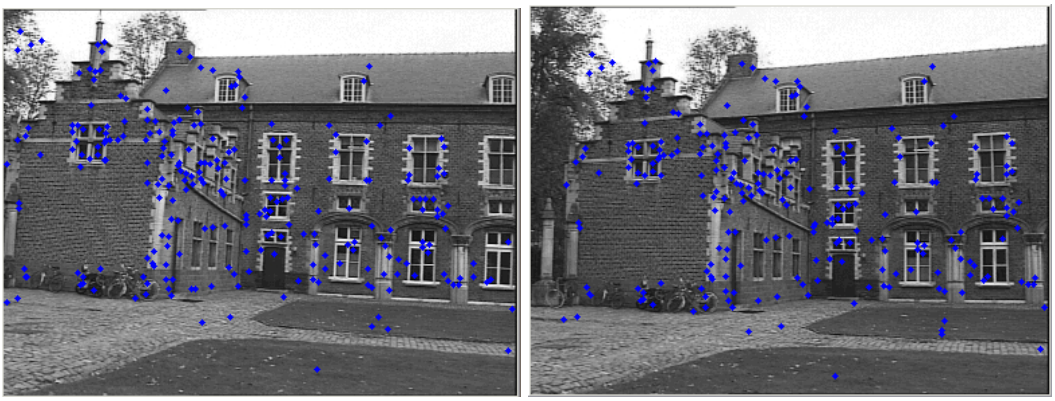
3. Remove repeated matches from the list.

Experimental Results

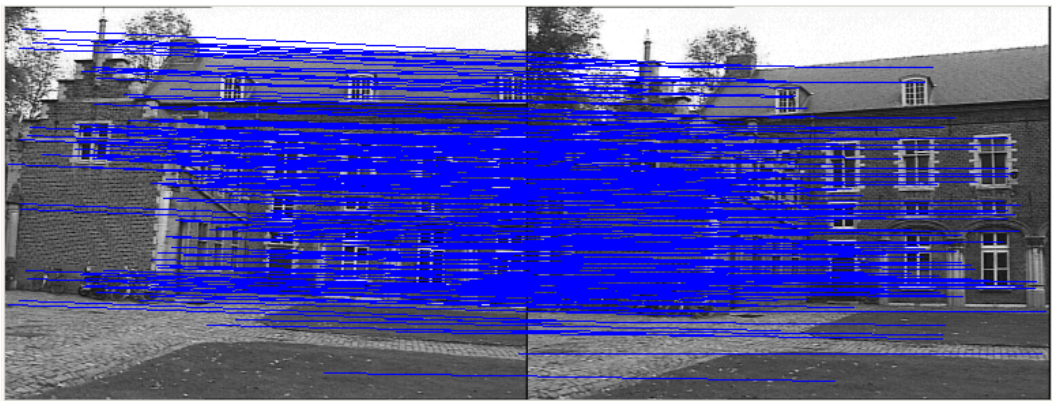
The method of Scale Invariant Feature Transform is applied to two images from the *Castle* sequence to detect features and obtain matches. The results are presented in Figure 2.8. In (a) and (b) the features detected on the first and second images are shown. The matches obtained from these features are represented as lines drawn from the first image to the second in (c), whereas the figure in (d) shows the displacements of the feature locations between the first and the second images.

2.2.2 Harris Corner Detector

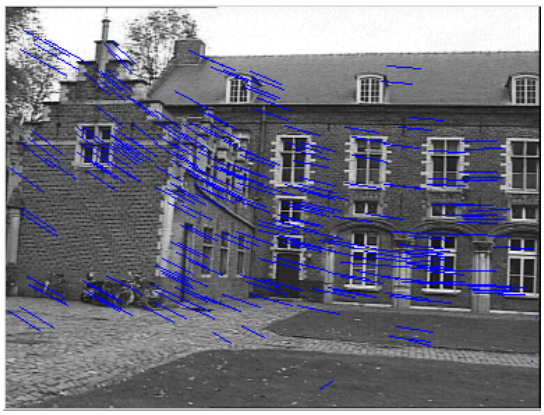
In 1988, Harris and Stephens developed a combined corner and edge detector that was to become one of the most popular feature detection tools in computer vision [3]. Being a widely used and accepted feature detection algorithm, Harris corner detector is regarded as a reference in this thesis to evaluate the performance of SIFT. In this section, some basic information about Harris corner detector is given. The performances of SIFT, Harris corner detector, and KLT tracker [20] are compared in Section 2.5.1.



(a) **(b)**



(c)



(d)

Figure 2.8: (a) Features detected in the first image **(b)** Features detected in the second image **(c)** Correspondences between the first and second images **(d)** Displacements of features from the first to the second images

The algorithm of Harris corner detector is based on the following observation: Given a windowed image region, consider the average change in image intensity as the window is shifted in various directions. If the windowed region is flat, the change will be small with respect to the shift in all directions; if the region is an edge, the change will be small when shifted along the edge, and high in the opposite direction; and finally, if the region is a corner, then the change will be high in all directions. The change in average intensity $E(x,y)$ with respect to the shift (x,y) is approximately

$$E(x,y) = \begin{bmatrix} x & y \end{bmatrix} \mathbf{M} \begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}, \tag{2.15}$$

$$\mathbf{M} = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix},$$

where W is the windowed image region under search, and I_x and I_y are the partial derivatives of the image in x and y directions. The term $E(x,y)$ in (2.15) is closely related to the image auto-correlation function, and the type of response at the windowed image region is associated with the two principal curvatures of the image auto-correlation. In order to have a corner in the searched region, the two principal curvatures must be close to each other.

Observing that the eigenvalues of **M**, α and β, are proportional to the two principal curvatures [3], the following function R can be treated as a cornerness measure:

$$R = \mathrm{Det}(\mathbf{M}) - k \cdot \mathrm{Tr}^2(\mathbf{M}) \tag{2.16}$$

where k is a constant, and Det(**M**) and Tr(**M**) are respectively the determinant and trace of the matrix **M**:

$$\begin{aligned} \mathrm{Det}(\mathbf{M}) &= \alpha\beta = (\sum_{W} I_x^2)(\sum_{W} I_y^2) - (\sum_{W} I_x I_y)^2 \\ \mathrm{Tr}(\mathbf{M}) &= \alpha + \beta = \sum_{W} I_x^2 + \sum_{W} I_y^2 \end{aligned} \tag{2.17}$$

The function *R*, which takes high values when the two eigenvalues are comparable to each other and low values when they are disproportional, provides a practical mechanism for the detection of corners. If the value of *R* at a pixel is the maximum among its 8 closest neighbours and also positive, then that pixel is accepted as a corner.

Performing feature detection by Harris corner detector as explained, feature matching can be achieved by comparing image intensity correlations of image patches taken around corner locations [9].

2.3 Epipolar Geometry Estimation

In order to obtain the sparse 3D structure of the scene from two views, the stage following feature detection and matching is the estimation of the epipolar geometry. Epipolar geometry is a geometric relation between two views, and is represented with the F-matrix.

In this thesis, F-matrix is computed by using correspondence pairs. There are several works addressing the problem of estimating the epipolar geometry from point correspondences [1],[10],[11]. 8-point algorithm [10],[11] is a method for fundamental matrix computation from eight matches, whereas 7-point algorithm [1] is a similar tool that estimates the F-matrix from 7-correspondences.

Another aspect of F-matrix computation from correspondence pairs is the problem of finding a suitable sampling strategy for forming sample sets from a large set of correspondences. The input to the epipolar geometry estimation stage is a set of

correspondences with many matches. However, the input set may also contain mismatches as well as correct matches, which raises the need for an efficient algorithm to choose good sample sets with the required number of correspondences from the input set, to be given to the F-matrix computation stage. RANSAC [12] is a widely used tool used for this purpose, which is based on computing models from randomly chosen sample sets until the correct model is found. On the other hand, PROSAC [13] is a modified version of RANSAC, which forms the sample sets not randomly but with respect to the quality of the matches.

Before discussing the estimation of the F-matrix from correspondences, a brief review of some basic concepts from two-view geometry is presented. An overview of epipolar geometry [1] is given in Section 2.3.1. The problem of computing the F-matrix from correspondences, and the 7-point and 8-point algorithms are discussed in Section 2.3.2, whereas the studied methods for sampling the set of correspondences efficiently, i.e. RANSAC and PROSAC, are explained in Section 2.3.3.

2.3.1 An Overview of Epipolar Geometry

Consider a two-camera setup such as the one illustrated in Figure 2.9. Let X be a 3D scene point, and C and C' be the centers of the first and second cameras, respectively. Then, the image of X in the first image plane will be the image point x, which is at the intersection of the line passing through X and C with the first image plane. The image of X on the second image plane, x', is obtained similarly. The scene point X, its projections x and x', and the camera centers C and C' are coplanar, and they lie in the plane π, which is called the *epipolar plane*. Obviously, an epipolar plane can be uniquely identified by a scene point and the camera centers, therefore, different epipolar planes are obtained if the position of the scene point is changed.

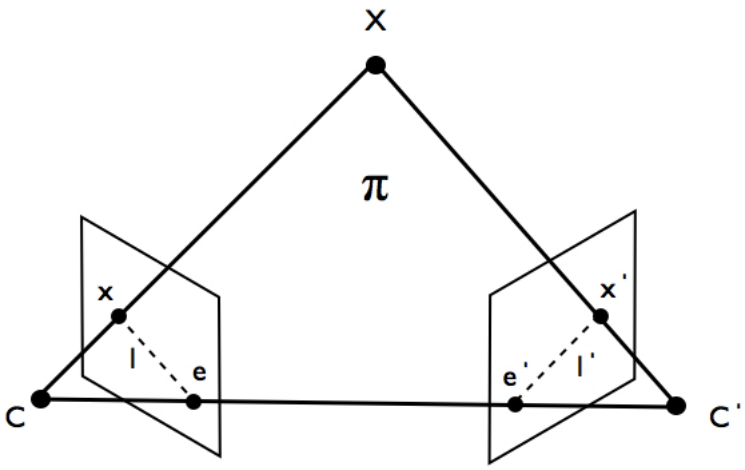


Figure 2.9: Illustration of a two-camera setup

The line connecting the two camera centers C and C' is called the *baseline*. The points e and e', which are located at the intersections of the baseline and the image planes, are called *epipoles*. Epipoles are unique for a given setup, and are independent of the scene structure.

The epipolar plane intersects each of the image planes along a line, and this line is called the *epipolar line*. In the illustration in Figure 2.9, the epipolar line corresponding to the scene point X in the first image plane is the line l (drawn dashed), which passes through x and e. The other epipolar line, l', is shown similarly on the second image plane. Different scene points (X_a, X_b,…) will in general be projected to different image points (x_a, x_b,… and x_a', x_b',..) on the image planes, and therefore will define different epipolar lines (l_a, l_b,… and l_a', l_b',…). However, all epipolar lines in the first image plane will be passing through the epipole e, and those in the second image plane through e'. Likewise, different epipolar planes (π_a, π_b,…) induced by different scene points (X_a, X_b,…) will all be intersecting along the baseline.

Now suppose that an image point x on the first image plane is given together with the two-view setup, and consider the problem of determining the image point x' on the second image plane which is the correct match of the image point x, meaning that x and x' must be the projections of the same scene point. The problem is illustrated in Figure 2.10. As shown in the figure, is not possible to determine a unique scene point corresponding to the image point x, since there are infinitely many scene points such as X_a and X_b on the ray passing through the camera center C and the image point x, all of which would be projected on x. However, since all candidate scene points, such as X_a and X_b, are collinear, their projections on the second image, such as x_a' and x_b', lie on the same line as well, namely, *epipolar line*, l'. Therefore, the problem of finding the match of a point x on the first image plane can be solved up to the level of mapping x to the epipolar line l', where the correct match of x, the point x', lies on the epipolar line l'.

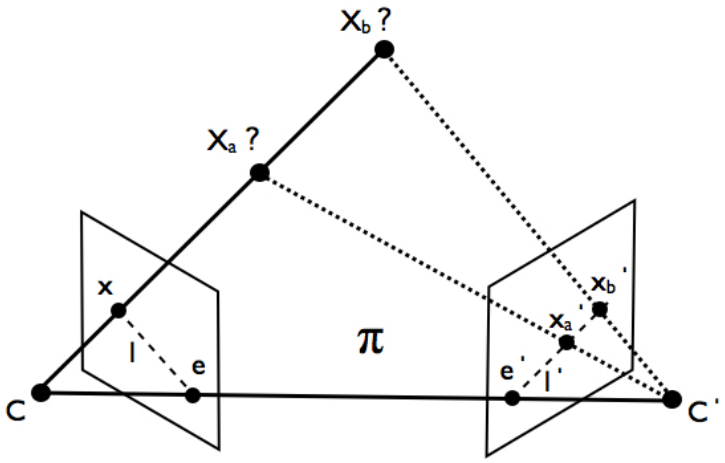


Figure 2.10: The problem of finding the match of x on the second image

It is possible to write the relation $x \rightarrow l'$, which maps every point in the first image to its corresponding epipolar line in the second image, in a matrix equation form [1]:

$$\mathbf{l}' = \mathbf{F}\mathbf{x}\,, \tag{2.18}$$

where **x** and **l'** are 3×1 matrices representing the point x and the line l' in 2D homogeneous coordinates, and **F** is a 3×3 matrix. The matrix **F** in Equation (2.18) is called the *fundamental matrix*, and constitutes an algebraic representation of the epipolar geometry between the two cameras.

The match of x in the second image, x', lies on the epipolar line l' corresponding to x, which can be formulated as

$$\mathbf{x}'^{\mathrm{T}}\mathbf{l}' = 0\,. \tag{2.19}$$

Combining (2.18) and (2.19), one obtains:

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0\,, \tag{2.20}$$

which is valid for all correspondences (x, x'). Therefore the fundamental matrix can be defined as a 3×3 matrix satisfying (2.20) for all correspondence pairs.

Some basic properties of the fundamental matrix are the following [1]:

- If **F** is the fundamental matrix relating the first image plane to the second, then $\mathbf{F}^{\mathrm{T}}$ is the fundamental matrix relating the second image plane to the first.
- Given any point x in the first image, its corresponding epipolar line l' in the second image satisfies $\mathbf{l}' = \mathbf{F}\mathbf{x}$.
- For any point x in the first image (other than the epipole e), the epipole of the second image e' lies on the epipolar line l' corresponding to x, which can be formulated by the equation $\mathbf{e}'^{\mathrm{T}}\mathbf{l}' = 0$. Combining this with the previous property, the result $(\mathbf{e}'^{\mathrm{T}}\mathbf{F})\mathbf{x} = 0$ is obtained. Since this is valid for

every **x**, one gets $\mathbf{e}'^{T}\mathbf{F}=0$, which means that **e'** is the left null vector of **F**. Similarly, it can also be shown that **e** is the right null vector of **F**, i.e. **Fe**=0.

- As **F** is a 3×3 matrix having one nontrivial null vector (from the previous property), it follows that **F** is of rank 2, and has 7 degrees of freedom.

2.3.2 Computation of the Fundamental Matrix from Correspondences

Computation of the fundamental matrix from a set of matches between two images is achieved by using the property of the F-matrix that $\mathbf{x}'^{T}\mathbf{F}\mathbf{x}=0$ for all matches (*x*,*x'*). Considering the entries of the F-matrix as unknowns and the point coordinates as known parameters, (2.20) can be put in the form of an equation system linear with respect to the F-matrix entries. Since each correspondence pair will define one equation, it is possible to solve the fundamental matrix linearly by using a sufficient number of correspondence pairs.

The 7-point and 8-point algorithms [1],[10],[11] are two methods that provide a solution for estimating the fundamental matrix from correspondences. The two algorithms are similar in the sense that both are based on solving a linear equation system constructed from the matches as explained. However, the basic difference between these methods is that the fundamental matrix is implicitly required to be rank-2 in the 7-point algorithm, whereas the immediate output of the 8-point algorithm is not necessarily rank-2 and needs to be converted to the closest rank-2 matrix to be a valid fundamental matrix.

8-point algorithm has been originally developed by Longuet-Higgins for computing the essential matrix with calibrated cameras [10]. In [11], Hartley has shown that the normalization of point coordinates improves the performance of the 8-point algorithm significantly by reducing its susceptibility to noise.

The basic mathematical formulation for the computation of F-matrix from correspondences, which 8-point and 7-point algorithms both stem from, can be obtained in a simple way [1]: For a correspondence pair (x, x'), the image points x and x' can be expressed as column matrices in homogeneous coordinates as

$$\mathbf{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \qquad \mathbf{x}' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}, \tag{2.21}$$

where u and v are respectively the horizontal and vertical coordinates of the image point x; and u' and v' are those of x'.

Rewritten in an open form using homogeneous coordinates, the equation $\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$ is equivalent to

$$\begin{bmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v & 1 \end{bmatrix}\mathbf{f} = 0, \tag{2.22}$$

where $\mathbf{f}$ is the 9×1 matrix composed of the entries of $\mathbf{F}$ written in row-major order.

If there are n correspondence pairs, each of them will define an equation in the form of (2.22), which can be combined to yield the linear equation system

$$\mathbf{A}\mathbf{f} = \begin{bmatrix} u_1'u_1 & u_1'v_1 & u_1' & v_1'u_1 & v_1'v_1 & v_1' & u_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_n'u_n & u_n'v_n & u_n' & v_n'u_n & v_n'v_n & v_n' & u_n & v_n & 1 \end{bmatrix}\mathbf{f} = 0. \tag{2.23}$$

The fundamental matrix $\mathbf{F}$ is obtained by solving Equation (2.23) for $\mathbf{f}$. The rank of $\mathbf{A}$ and whether an exact solution exists depend on both the number of correspondences n, and the coordinate measurements, which will be explained when discussing the 7-point and the 8-point algorithms in the following sections.

2.3.2.1 Seven-Point Algorithm

The minimum number of correspondences required for determining the F-matrix from (2.23) is seven. 7-point algorithm is a procedure for computing the F-matrix from seven correspondences. When there are seven correspondences, the matrix **A** in (2.23) will have dimensions 7×9. In the most likely case, **A** will be of rank 7, and have a two-dimensional null space.

The null space of **A** is generated by two independent null vectors $\mathbf{f_1}$ and $\mathbf{f_2}$, and can be parameterized as $\alpha\mathbf{f_1} + (1-\alpha)\mathbf{f_2}$, where α is a scalar. This leads to the result that the sought fundamental matrix **F** will be of the form $\alpha\mathbf{F_1} + (1-\alpha)\mathbf{F_2}$, where $\mathbf{F_1}$ and $\mathbf{F_2}$ are the 3×3 matrices obtained by rearranging $\mathbf{f_1}$ and $\mathbf{f_2}$. Combining this result with the information that **F** is a rank-2 matrix, one can write

$$\det(\alpha\mathbf{F_1} + (1-\alpha)\mathbf{F_2}) = 0, \tag{2.24}$$

which is a third-degree polynomial equation in α, having either one or three real roots. Complex solutions are discarded if there are any, and each solution for α yields a solution for the fundamental matrix **F** as [1]:

$$\mathbf{F} = \alpha\mathbf{F_1} + (1-\alpha)\mathbf{F_2} \tag{2.25}$$

2.3.2.2 Eight-Point Algorithm

8-point algorithm is a method for computing the F-matrix from at least 8 correspondences. In order to have an exact unique solution for **f** up to scale in equation (2.23), the matrix **A** must be of rank 8. If **A** is rank-8, (2.23) can be solved easily for **f** by determining the null vector of **A**. When the equation system is constructed with 8 correspondences, the most probable case is to have a rank-8

matrix. However, 8-point algorithm allows the number of correspondences to be more than 8, as well, and in this case, **A** is not guaranteed to be rank-8 due to noise in correspondence coordinate measurements. Therefore, the algorithm follows a general procedure that can be applied to any $n\times 9$ matrix **A**, where $n \geq 8$.

If the linear system in (2.23) is constructed from n noisy correspondences for $n \geq 8$, **A** may have rank higher than 8, i.e. 9 as it has 9 columns. Therefore the system might not have an exact solution. In this case, a least squares solution is sought, which minimizes $\|\mathbf{Af}\|$ subject to the condition $\|\mathbf{f}\| = 1$. The solution to this minimization problem can be obtained by computing the singular value decomposition (SVD) of **A,** such that $\mathbf{A} = \mathbf{USV}^{\mathrm{T}}$, where **U** and **V** are orthogonal matrices, and **S** is a matrix containing the singular values of **A** on its diagonal entries. The least squares solution **f** to **Af**=0 is given by the column of **V** corresponding to the smallest singular value of **A**, which is usually the last column.

When the matrix **f** is computed as a solution to (2.23) as explained, the required fundamental matrix **F** can be obtained from **f** by rearranging it in its original 3×3 form. However, the fundamental matrix **F** estimated in this way need not be rank-2, as this has not been imposed as a constraint in the performed calculations. Therefore the obtained solution for **F** must be explicitly converted to the closest rank-2 matrix $\tilde{\mathbf{F}}$. This can be formulated as an optimization problem, where $\|\mathbf{F} - \tilde{\mathbf{F}}\|$ is minimized subject to $\det(\tilde{\mathbf{F}}) = 0$. Again, it is possible to solve this minimization problem by singular value decomposition. The original matrix **F** is factorized as

$$\mathbf{F} = \mathbf{USV}^{\mathrm{T}} = \mathbf{U}\begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix}\mathbf{V}^{\mathbf{T}} \tag{2.26}$$

where the singular values of **F**, i.e. s_1, s_2 and s_3, are ordered as $s_1 \geq s_2 \geq s_3$. In order to reduce the rank, **S** is modified to get $\tilde{\mathbf{S}}$=diag(s_1, s_2, 0), and the rank-2 approximation of **F**, denoted by $\tilde{\mathbf{F}}$, is obtained as [1]

$$\tilde{\mathbf{F}} = \mathbf{U}\begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}\mathbf{V}^{\mathrm{T}}. \tag{2.27}$$

2.3.2.3 Normalization of Correspondence Coordinates

During the calculation of the fundamental matrix with 7-point and 8-point algorithms, the constructed equation system in (2.23) might become ill-conditioned due to noise in image point coordinate measurements. In [11], it has been shown that a simple normalization of coordinates, composed merely of translation and scaling, improves the conditioning of the problem substantially and increases the stability of the result.

During this normalization, the 2D image points are translated so that the centroid of points is moved to the origin. The translation is followed by a scaling so that the RMS of the distance of points to the origin becomes $\sqrt{2}$. This procedure is done for the features in both images. Denoting the mean horizontal and vertical coordinates of features by m_u and m_v, and the RMS distance to the centroid by d_{RMS}, the normalization applied to an image point can be represented as the linear transformation

$$\mathbf{x_n} = \mathbf{T}\mathbf{x}, \tag{2.28}$$

$$\mathbf{x_n} = \begin{bmatrix} u_n \\ v_n \\ 1 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \qquad \mathbf{T} = \begin{bmatrix} \frac{\sqrt{2}}{d_{RMS}} & 0 & -m_u \frac{\sqrt{2}}{d_{RMS}} \\ 0 & \frac{\sqrt{2}}{d_{RMS}} & -m_v \frac{\sqrt{2}}{d_{RMS}} \\ 0 & 0 & 1 \end{bmatrix},$$

where $\mathbf{x}$ and $\mathbf{x_n}$ are respectively the image point locations represented in homogeneous coordinates before and after normalization, and $\mathbf{T}$ is the transformation matrix.

After the fundamental matrix is computed with the normalized correspondences, it is denormalized to give the solution of the original problem. The overall procedures for obtaining the F-matrix from correspondences with 7-point and 8-point algorithms, including the normalization steps, are summarized as follows:

Algorithm Overview: 7-Point Algorithm

Input: Seven correspondence pairs (**x**, **x'**) between two images
Output: The fundamental matrix relating the two images

1. Transform the coordinates of the points in both images (**x** and **x'**) with respect to the transformation described in (2.28) to obtain the normalized coordinates $\mathbf{x_n}$ and $\mathbf{x_n'}$.

$\mathbf{x_n} = \mathbf{T}\mathbf{x}$
$\mathbf{x_n'} = \mathbf{T'}\mathbf{x'}$

2. Construct the linear equation system in (2.23), and obtain the two null vectors $\mathbf{f_1}$ and $\mathbf{f_2}$ of matrix **A**.

3. Determine matrices $\mathbf{F_1}$ and $\mathbf{F_2}$ by rearranging the null vectors $\mathbf{f_1}$ and $\mathbf{f_2}$.

4. Obtain the polynomial equation in (2.24) using $\mathbf{F}_1$ and $\mathbf{F}_2$, and solve for α.

5. Determine the normalized fundamental matrix $\mathbf{F}_n$ corresponding to each solution for α from (2.25).

6. Denormalize the normalized fundamental matrix $\mathbf{F}_n$ in order to obtain the original fundamental matrix **F** by the back-transformation

$$\mathbf{F} = \mathbf{T}'^{T} \mathbf{F}_n \mathbf{T}.$$

Algorithm Overview: 8-Point Algorithm

Input: Eight or more correspondence pairs (**x**, **x'**) between two images
Output: The fundamental matrix relating the two images

1. Transform the coordinates of the points in both images (**x** and **x'**) with respect to the transformation described in (2.28) to obtain the normalized coordinates $\mathbf{x}_n$ and $\mathbf{x}_n'$.

$$\mathbf{x}_n = \mathbf{T}\mathbf{x}$$
$$\mathbf{x}_n' = \mathbf{T}'\mathbf{x}'$$

2. Construct the linear equation system in (2.23), and determine the least squares solution for **f** by SVD.

3. Determine the normalized fundamental matrix $\mathbf{F}_n$ by rearranging **f**.

4. Compute the rank 2 approximation $\tilde{\mathbf{F}}_n$ of $\mathbf{F}_n$ from (2.26) and (2.27).

5. Denormalize the normalized fundamental matrix $\tilde{\mathbf{F}}_n$ in order to obtain the original fundamental matrix **F** by the back-transformation

$$F = T'^{T} \tilde{F}_n \; T.$$

2.3.3 Sampling Strategies in Fundamental Matrix Estimation

The 7-point and 8-point algorithms, which compute the fundamental matrix from seven and at least eight correspondences respectively, have been discussed in the previous section. However, the problem of estimating the epipolar geometry from a set of correspondences raises the question of which correspondences to use. For example, 7-point algorithm takes only seven correspondences as input; therefore, one must determine seven matches out of a large set of matches. 8-point algorithm allows more than eight points; however, it is not preferable to compute the F-matrix directly from all matches, since the input set of matches usually contains mismatches as well. In the case that mismatches constitute a considerable percentage of all matches, the least squares solution for the F-matrix will be significantly contaminated by the incorrect data. These points lead to the necessity for a tool that will determine the correct matches while estimating the valid model efficiently at the same time.

Random Sample Consensus (RANSAC) method was developed by Fischler and Bolles with this idea in 1981 [12] and has become a very popular tool to be used in such applications. In the following years, RANSAC has been the inspiration to many similar algorithms, such as PROSAC [13], DEGENSAC [14] and MLESAC [15], all of which are derived from RANSAC through some modifications.

The epipolar geometry estimation stage of this thesis focuses on RANSAC and PROSAC. The algorithm of RANSAC is explained in Section 2.3.3.1, and PROSAC is discussed in Section 2.3.3.2. In order to compare the performances of the two algorithms experimentally, several tests have been conducted, and the results of these tests are presented in Section 2.5.2.

2.3.3.1 Random Sample Consensus (RANSAC)

RANSAC (Random Sample Consensus) is an algorithm for fitting a model to experimental data containing considerable gross errors [12]. Unlike classical methods, such as least squares, which do not detect the erroneous data, RANSAC is a *hypothesize-and-verify* type of algorithm that estimates the model from a minimal subset of the data, and in this way tries to find the model consistent with as much of the data as possible.

In the case of estimating the fundamental matrix from a set of correspondences, the algorithmic procedure carried out is the following: A minimal sample set is chosen randomly from the whole set of matches, i.e. a sample set of 7 correspondences is formed if 7-point algorithm is to be applied, whereas 8 correspondences are chosen in the case of 8-point algorithm. A fundamental matrix is computed from this sample set, and then the validity of the estimated F-matrix is tested by checking the proportion of consistency with the estimated model among the whole input set. The matches consistent with the estimated F-matrix are called *inliers*, and the disagreeing matches are denoted as *outliers*. Then another iteration is made with another randomly chosen sample set, and this procedure is repeated until an enough number of iterations are made. The estimation for the F-matrix is updated whenever the currently tested model yields the highest inlier ratio among the models tried up to that moment.

The criterion used for determining whether a correspondence is consistent with a fundamental matrix is the *Sampson error*. Sampson error is a close approximation to geometric error and was first used by Sampson for conic fitting [16]. In the problem of fundamental matrix estimation, Sampson error can be defined as a function of a correspondence (x, x') and a fundamental matrix $\mathbf{F}$, and is in the form [1]

$$S = \frac{(\mathbf{x'}^{\mathrm{T}}\mathbf{F}\mathbf{x})^2}{(\mathbf{F}\mathbf{x})_1^2 + (\mathbf{F}\mathbf{x})_2^2 + (\mathbf{F}^{\mathrm{T}}\mathbf{x'})_1^2 + (\mathbf{F}^{\mathrm{T}}\mathbf{x'})_2^2} \quad , \tag{2.29}$$

where **x** and **x'** are column matrices representing the correspondence in homogeneous coordinates, and the expressions $(.)_1$ and $(.)_2$ denote respectively the first and second entries of a column matrix. Sampson error is a measure of the compatibility of a match with a fundamental matrix, where the error decreases as the match agrees with the model better. If the correspondence is perfectly consistent with **F**, the term $\mathbf{x'}^{\mathrm{T}}\mathbf{F}\mathbf{x}$ in the numerator of the expression in (2.29) vanishes; therefore, the Sampson error is zero. If the Sampson error computed for a match and a fundamental matrix is below some threshold, then the match is determined to be an inlier with respect to the model defined by the fundamental matrix, and an outlier otherwise.

In addition to adopting the approach of choosing random minimal samples, RANSAC also incorporates an efficient termination mechanism. The number of iterations to be made is determined adaptively depending on the progress of the model estimation procedure. The input correspondence set contains both correct and false matches, which can be regarded as inliers and outliers with respect to the correct model. However, it is never possible to be absolutely sure that the correct model is obtained. Therefore, the termination decision is made in RANSAC to ensure with some probability that the best model ever estimated is the correct one.

Denoting the number of inliers by *I,* and the number of all matches by *N*; the probability that a randomly chosen match is an inlier is $P_I = I/N$. If the sample sets contain *m* correspondences, the probability of having all inliers in a sample set is approximately equal to P_I^m. Therefore, the probability that at least one of the sample sets tested until the k^{th} iteration is free of outliers is $1-(1-P_I^m)^k$, which

can be regarded as the probability of having found the correct model. Setting this probability to a predetermined constant η, one has

$$\eta = 1 - (1 - P_I^m)^k$$

Hence, the number of iterations k_{MAX} to be made to ensure with a probability η that the correct model is estimated is

$$k_{MAX} = \frac{\log(1-\eta)}{\log(1-P_I^m)}. \tag{2.30}$$

The upper bound for the number of iterations k_{MAX} is updated, whenever the model is updated. When the number of iterations made reaches or exceeds k_{MAX}, the algorithm terminates. It is possible to refine the final model further by determining the inliers with respect to the final model and recalculating the model from all inliers in an iterative manner.

A possible improvement on RANSAC is the inclusion of the *bucketing* technique to the algorithm [17]. Since the sample sets are formed randomly, it may sometimes happen that the sample matches are confined to a restricted region of the image plane, which might reduce the reliability of the result. In order to prevent this, image planes are divided into equal regions, called *buckets*, and the members of a sample set are chosen such that no two of them belong to the same bucket in either image.

The pseudo-code for RANSAC is the following:

Algorithm Overview: Fundamental Matrix Estimation with RANSAC

Input: A set of correspondences between two images
Output: The fundamental matrix relating the two images

1. Initializations:

a. Set the number of iterations made to 0 (k=0).

b. Set the upper bound for the number of iterations to an initial value (k_{MAX}=M).

c. Set the highest number of inliers to 0 (I_{MAX}=0).

2. While (k<k_{MAX})

a. Choose a random sample set.

b. Compute the fundamental matrix from the sample set.

c. Determine the number of inliers I by calculating the Sampson error for each match and classifying the match as an inlier if the error is below some predetermined threshold.

d. If (I>I_{MAX})

i. Update the best model (F-matrix) obtained so far.

ii. Set I_{MAX}=I.

iii. Update k_{MAX} with respect to (2.30).

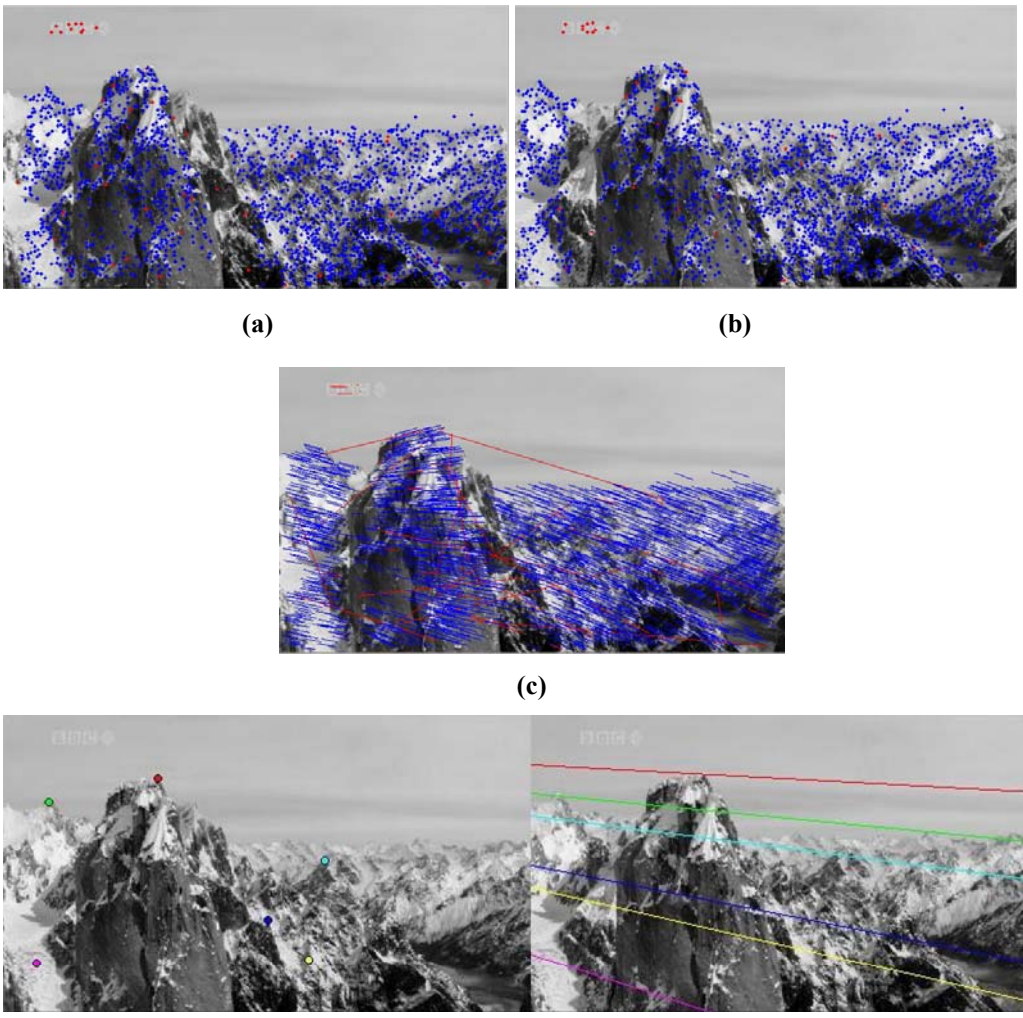
e. Increment the number of iterations (k=k+1).

3. Refine the final model: Repeat steps **a**-**b** until the model converges or an enough number of iterations are made.

a. Determine the inliers consistent with the model.

b. Recalculate the fundamental matrix from all inliers (8-point algorithm is to be used in this step since the number of inliers is usually more than 8).

The RANSAC algorithm is tested on two views of a scene from the documentary video *Planet Earth*, to estimate the epipolar geometry between the views. The results are presented in Figure 2.11. In (a) and (b), the first and second views are shown, together with the detected (and matched) SIFT features. The features shown with blue dots are determined as inliers by RANSAC, whereas the red dots indicate outliers. In (c), the disparity vectors showing the displacement of the features between the first and the second views are drawn, where the inlier and outlier matches are again indicated with blue and red colours respectively. The



(a) **(b)**

(c)

(d)

Figure 2.11: Epipolar geometry estimation with RANSAC: **(a)** First image **(b)** Second image **(c)** Displacement vectors of features **(d)** Epipolar lines in the second image and the corresponding points in the first image

epipolar lines on the second image are drawn in (d), where the line with each colour corresponds to the point in the first image having the same colour.

The results given in Figure 2.11 prove the performance of RANSAC for detecting and discarding the outliers. Most of the outlier matches can be easily identified in (c) as false lines in discord with the general flow of disparity vectors; and it can be observed that RANSAC efficiently discriminates the outliers from inliers. Moreover, the matches detected on the TV channel logo, which are not part of the scene and thus inconsistent with the overall geometry, are also labeled successfully as outliers. The epipolar lines in (d) are consistent with the points marked in the first image, i.e. the matches of the selected points lie on the corresponding epipolar line, and the directions of the lines are also as expected, considering the camera motion between the two views.

2.3.3.2 Progressive Sample Consensus (PROSAC)

Progressive Sample Consensus (PROSAC) [13] is an algorithm proposed by Chum and Matas for estimating the model between two views from image correspondences. PROSAC is a modified version of RANSAC that is devised to achieve computational savings over RANSAC, while retaining its performance as far as the quality of the obtained model is concerned.

PROSAC orders input correspondences due to some quality (similarity) measure, and forms sample sets in a manner that favours the correspondences with better scores, whereas RANSAC draws samples randomly. The approach of giving priority to matches with high similarity scores in PROSAC relies on the assumption that the similarity function of a match provides a good prediction about its correctness.

The quality measure used in ranking correspondences is the similarity function assigned to correspondences in the feature matching stage. The similarity function can be defined in various ways depending on the type of feature matcher: The intensity correlation around features may be an appropriate similarity measure if feature detection is achieved with Harris corner detector, whereas the most suitable quality measure in SIFT is the distance ratio, which is explained in Section 2.2.1.

The procedure applied to estimate the model with PROSAC can be summarized as follows: Consider an input set of N correspondences denoted by U_N, where the tentative matches in U_N are sorted with respect to a quality function q, such that for any $u_i, u_j \in U_N$, $q(u_i) > q(u_j)$ if $i < j$. Since feature detection and matching is achieved with SIFT in this thesis, the quality function q is taken as $1-(d_1/d_2)$, where d_1 and d_2 are respectively the distances of the descriptor vectors of the first and second closest features in the second image to the descriptor vector of the feature in the first image.

The subset of U_N consisting of the top n correspondences with the highest quality is denoted by U_n, and the sample sets consisting of m samples are denoted by M, where m depends on the fundamental matrix estimation method. PROSAC is similar to RANSAC in the sense that samples are drawn randomly from a hypothesis generation set and the correct model is obtained after trying a series of models computed from different sample sets. The hypothesis generation set is the whole set of matches in RANSAC, i.e. U_N, whereas in PROSAC it is a growing set that initially consists of the top m correspondences and is gradually extended by adding new correspondences one by one in quality order throughout the iterations.

The growth of hypothesis generation set is controlled with respect to the number of iterations made via a growth function $g(k)$, which determines the size of the

hypothesis generation set n, as a function of the iteration number k. An example growth function is proposed in [13]; however, it is possible to define alternative growth functions, depending on the reliability of the quality function as a measure of the probability of the correctness of a match.

Once the growth function g is defined, sample sets can be formed according to the relation

$$M_k = \left\{u_{g(k)}\right\} \cup M_k^{'}, \tag{2.31}$$

where M_k stands for the sample set chosen at the k^{th} iteration, $u_{g(k)}$ is the correspondence with the quality order $g(k)$, and M_k' is a subset of $U_{g(k)\text{-}1}$ formed by choosing m-1 samples from $U_{g(k)\text{-}1}$ randomly. The intention in constituting sample sets in this way is to give more chance to top-ranking correspondences when estimating the model, and try other correspondences in quality order if model estimation fails.

Forming sample sets as explained, the rest of the algorithm is similar to RANSAC. Fundamental matrix is computed from the chosen sample set, and the validity of the computed model is tested by checking if the correspondences in the hypothesis generation set are consistent with the model. The consistency of a correspondence with the model is again measured with the Sampson error, which is formulated in (2.23). A series of models are estimated and tested in a cycle, until some termination criteria are satisfied. The best model estimated until termination is the output of the algorithm.

The termination decision in PROSAC is given with respect to two criteria; *non-randomness* and *maximality*. These two concepts are explained as follows [13]:

1) Non-randomness: In order to terminate, the number of inliers i_n within the hypothesis generation set U_n must satisfy the condition that the probability of classifying i_n of the correspondences as inliers among n correspondences by chance even if the model is incorrect is smaller than some threshold ψ. This constraint is imposed in order to prevent the algorithm from ending up with an incorrect model supported by fake inliers that are actually outliers supporting the model by chance. The probability of having i inliers among n correspondences for an incorrect model is

$$P^R{}_n(i) = \beta^{i-m}(1-\beta)^{n-i+m}\binom{n-m}{i-m}, \tag{2.32}$$

where β denotes the probability that an incorrect model is supported by a randomly chosen correspondence not in the sample set inducing that model. β can be estimated adaptively from the statistics obtained during the runtime of the algorithm. In order to check the non-randomness condition, for the size n of the hypothesis generation set, the minimum number of inliers $i_n^{\min}$ is calculated, where the probability of having a support of size $i_n^{\min}$ by chance due to an incorrect model is less than ψ. The term $i_n^{\min}$ is given by

$$i_n^{\min} = \min\left\{ j \mid \sum_{i=j}^{n} P^R{}_n(i) < \psi \right\}, \tag{2.33}$$

where the function $P^R{}_n(i)$ is as defined in (2.32). In order to satisfy the non-randomness condition for termination, the number of inliers i_n must satisfy $i_n \geq i_n^{\min}$.

2) Maximality: The maximality condition is a constraint on the number of iterations. A sufficient number of iterations must be done until termination to

ensure that the probability of having found the correct model is higher than some predetermined threshold. The maximality condition is actually the only termination criterion in RANSAC, and the number of iterations required for termination is given in (2.30). However, when calculating the required number of iterations in PROSAC, the ratio of inliers is evaluated not within the whole set of correspondences U_N as in RANSAC, but within the hypothesis generation set U_n.

The algorithm terminates only if both the non-randomness and the maximality conditions are satisfied. The whole procedure for estimating the fundamental matrix with PROSAC is summarized as follows:

Algorithm Overview: Fundamental Matrix Estimation with PROSAC

Input: A set of correspondences between two images sorted with respect to a quality function
Output: The fundamental matrix relating the two images

1. Initializations:
   a. Set the number of iterations made to 0 (k=0).
   b. Set the number of inliers to 0 (i_n=0).
   c. Set the upper bound for the number of iterations to an initial value (k_{MAX}=M).
   d. Set the size of the hypothesis generation set to the sample size (n=m).
   e. Set the minimum number of inliers to the sample size (i_n^{min}=m).

2. While (k<k_{MAX} or i_n<i_n^{min})
   a. Increment the number of iterations (k=k+1).
   b. If the size of the hypothesis generation set requires an update due to the growth function
      i. Update n=g(k).
      ii. Update i_n^{min} according to (2.33).

c. Take u_n and m-1 random samples from U_{n-1} to form the sample set (except for the case n=m, where the sample set is the hypothesis generation set itself).

d. Compute the fundamental matrix from the sample set.

e. Determine the number of inliers i_n.

f. If the estimated model is the best model so far

i. Update the model.

ii. Update k_{MAX} with respect to (2.30).

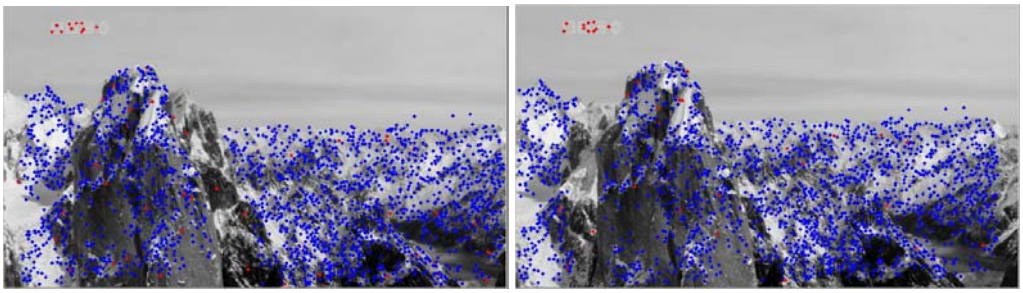
3. Refine the final model: Repeat steps **a**-**b** until the model converges or an enough number of iterations are made.

a. Determine the inliers consistent with the model.

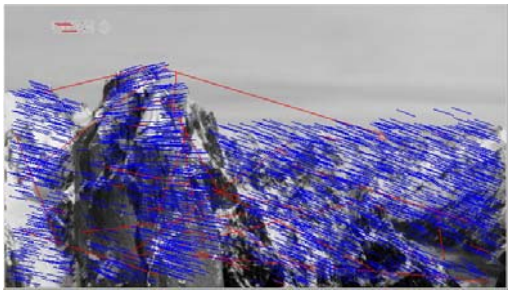
b. Recalculate the fundamental matrix from all inliers.

The performance of PROSAC is also tested on the *Planet Earth* sequence, the results of which are given in Figure 2.12. The features detected on the first and second images are shown respectively in (a) and (b). Displacement vectors of features between the two images are drawn in (c), and the epipolar lines on the second image are drawn in (d). Blue colour indicates inliers and red colour indicates outliers in (a), (b) and (c).

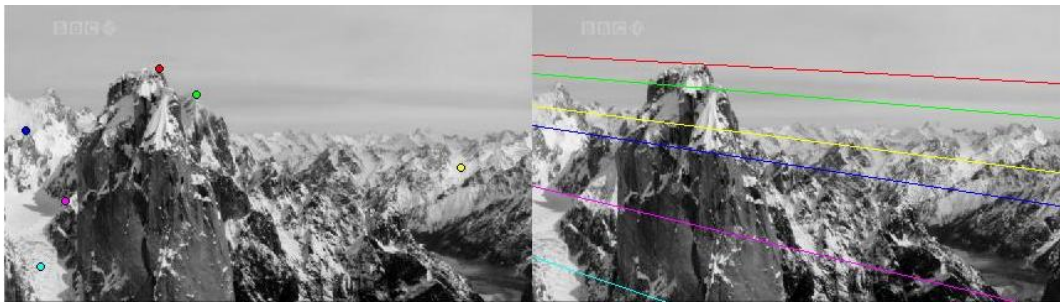
The data used in this experiment is the same as that of the experiment done with RANSAC, whose results are presented in Figure 2.11. The plots given in Figure 2.12 reveal that PROSAC is capable of discarding outliers successfully and estimating the correct model. The similarity of the results obtained with RANSAC and PROSAC is also impressive, which suggests that PROSAC performs as well as RANSAC in computing the correct model from a set of data contaminated with outliers. A more comprehensive comparison between PROSAC and RANSAC is made in Section 2.5.2.



(a) **(b)**



(c)



(d)

Figure 2.12: Epipolar geometry estimation with PROSAC: **(a)** First image **(b)** Second image **(c)** Displacement vectors of features **(d)** Epipolar lines on the second image and the corresponding points in the first image

2.4 Reconstruction of Cameras and Structure

Following the discussions on feature detection and epipolar geometry estimation, the last stage to be examined regarding the two-view reconstruction problem is the reconstruction of the cameras and structure, which is the recovery of the 3D organization of the scene and the camera poses. In other words, the reconstruction problem is the determination of the 3D locations of scene points together with the positions and orientations of cameras, so that the solution explains the measurements on 2D correspondence coordinates.

Once fundamental matrix is estimated, it is possible to recover the camera poses, i.e. camera projection matrices, up to a projective transformation directly from the fundamental matrix. The structure can then be reconstructed via triangulation, again up to a projective ambiguity, which means that it is possible to reconstruct the scene projectively, even if the two-camera system is uncalibrated. If internal calibration information is available, it is also possible to stratify the projective structure to get the metric reconstruction of the scene.

In this thesis, the 3D metric reconstruction problem is solved in the explained manner, i.e. the cameras and scene structure are first reconstructed up to a projective transformation, and then upgraded to metric through the internal calibration information. However, it is also possible to handle the same problem in different ways; for example, an alternative approach [9] is calculating the essential matrix from the fundamental and the internal calibration matrices, and then decomposing the essential matrix to obtain the rotation and translation parameters, which define the metric reconstruction directly. Instead of adopting such an approach, attaining metric reconstruction through projective is preferred in this thesis. This is due to the expectation of avoiding the error accumulating at the cost of an ideally metric representation, while new views are being added to the system in the multi-view reconstruction stage.

This section is organized as follows: In Section 2.4.1 some background information about projective geometry is presented in order to clarify the concepts referred to in the proceeding discussions. In Section 2.4.2 the projective reconstruction problem is discussed; and the procedure of upgrading the projective reconstruction to metric is explained in Section 2.4.3.

2.4.1 An Overview of Projective Geometry

A camera is defined as a mapping between the 3D world and the 2D image [1]. This mapping can be expressed via the relation

$$\mathbf{x} = \mathbf{PX} \tag{2.34}$$

where **P** is the 3×4 *projection matrix* defining the camera, **X** is a 3D scene point, and **x** is the 2D projection of **X** on the image plane of the camera. Both **X** and **x** are represented in homogeneous coordinates. Figure 2.13 illustrates the projection of a 3D scene point onto an image plane via a pinhole camera.

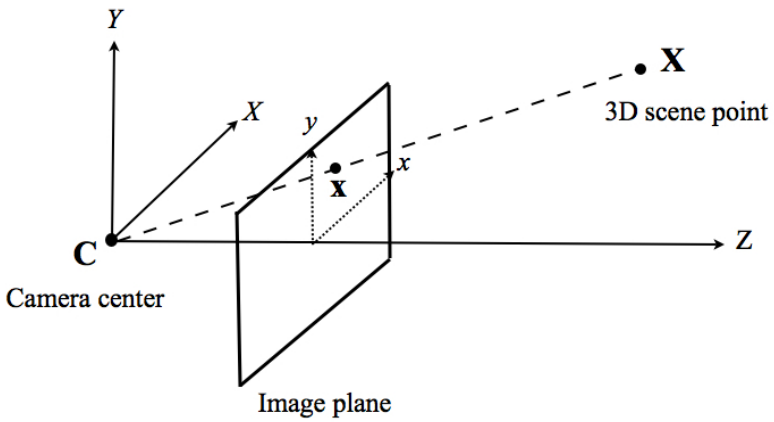


Figure 2.13: Projection of the 3D scene point **X** onto the image plane of the pinhole camera with center **C**, to obtain the 2D image point **x**.

Homogeneous coordinates of a point are obtained by adding a scale entry at the end of usual coordinates. For example, a 3D point with coordinates (X, Y, Z) can be expressed in homogeneous coordinates as (X, Y, Z, 1), or equivalently as (kX, kY, kZ, k) for any scalar k. Similarly a 2D point with coordinates (x, y) is expressed as (cx, cy, c) in homogeneous coordinates. Therefore, (2.34) can be rewritten as:

$$\begin{bmatrix} cx \\ cy \\ c \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix}$$

Projective Cameras

The most general camera form is the projective camera. A projective camera maps 3D world points to 2D image points via the relation **x**=**PX**. A general projective camera is represented by an arbitrary 3×4 matrix of rank 3.

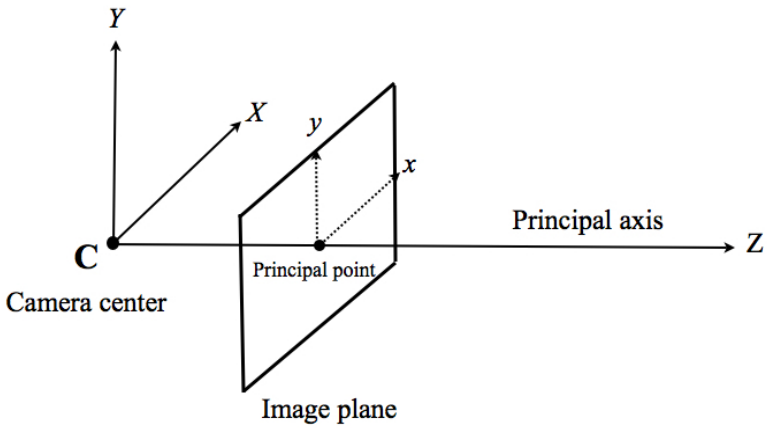


Figure 2.14: Camera center, image plane, principal point and the principal axis

Since **P** is a rank-3 matrix with 3 rows and 4 columns, it has a one-dimensional null space. Moreover, the vector **C** generating the null space of **P**, such that **PC=0**, is the *camera center* of **P** represented in homogeneous coordinates.

The *principal axis* is the line passing through the camera center **C** with direction perpendicular to the image plane. The intersection of the principal axis with the image plane is the *principal point*.

Pinhole Camera Model

A pinhole camera is a special form of projective camera that constitutes a simple model for most commercial cameras. The pinhole camera model maps a 3D point **X** to the 2D point **x** on the image plane, which is at the intersection of the line joining the camera center **C** to the scene point **X**. This mapping is illustrated in Figure 2.13.

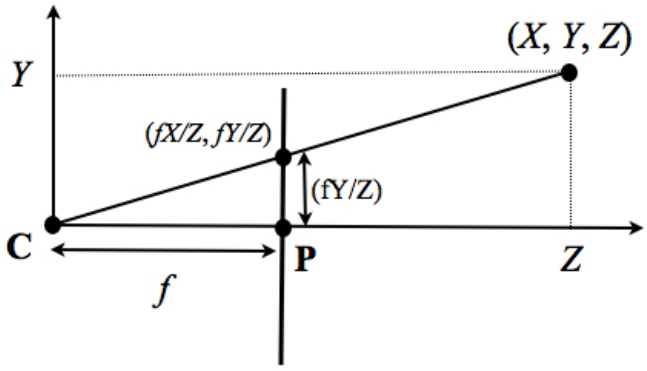


Figure 2.15: Projection of the scene point (*X*, *Y*, *Z*) to the image point (*fX*/*Z*, *fY*/*Z*) with a pinhole camera

Denoting the *focal length* of the camera by *f*, which is defined as the distance between the camera center and the image plane, it is easy to establish the mapping between the 3D point (*X*, *Y*, *Z*) and its 2D projection (*fX*/*Z*, *fY*/*Z*) on the image

using basic geometry rules, where the 3D coordinates are represented with respect to the camera coordinate frame taking the camera center as origin, and the 2D coordinates are represented with respect to image coordinate frame. The projection of a 3D point to the image plane is explained in Figure 2.15.

The mapping $(X, Y, Z) \rightarrow (fX/Z, fY/Z)$ assumes that the principal point is the origin of the image plane. In the general case where the principal point has coordinates (p_x, p_y), the mapping can be generalized as $(X, Y, Z) \rightarrow (fX/Z+p_x,\ fY/Z+p_y)$. This can be expressed in a matrix equation of form **x**=**PX** as

$$\begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.35}$$

where the scene point (X, Y, Z) and the image point $(fX/Z+p_x,\ fY/Z+p_y)$ are expressed in homogeneous coordinates as **X** and **x**. Equation (2.35) can be rewritten as **x**=**K**[**I** | **0**]**X**, where **K** is

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.36}$$

The matrix **K** is called the *camera calibration matrix*. In (2.35), the scene point **X** is represented with respect to the camera coordinate frame. However, sometimes it is necessary to represent **X** with respect to a different world coordinate frame. In the general case, the world coordinate frame and the camera coordinate frame are related via a rotation and translation. Therefore, the equation **x**=**K**[**I** | **0**]**X** is modified as follows so that **X** is represented with respect to the world coordinate frame instead of the camera coordinate frame:

$$\mathbf{x}=\mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} \tag{2.37}$$

In (2.37), **K** is the camera calibration matrix, and **R** and **t** are the rotation matrix and the translation vector defining the conversion between the camera and the world coordinate frames.

To summarize, in addition to inheriting the generic properties of the projective camera, a pinhole camera must also be in the special form described in (2.37).

Projective and Metric Reconstructions

The purpose of two-view reconstruction is, given a set of correspondences $\{(x_i, x_i')\}$, to determine the projection matrices **P** and **P'** corresponding to the two cameras, and 3D scene points $\{\mathbf{X}_i\}$ satisfying $\mathbf{x}_i=\mathbf{P}\mathbf{X}_i$ and $\mathbf{x}_i'=\mathbf{P}'\mathbf{X}_i$ for every i.

In any reconstruction problem, where the scene structure and cameras are to be retrieved from image correspondences $\{(x_i, x_i')\}$, there exists a true reconstruction $\bar{\mathbf{P}}$, $\bar{\mathbf{P}}'$ and $\{\bar{\mathbf{X}}_i\}$, and it is possible to obtain the true reconstruction up to some type of transformation, such as a *similarity transformation*, *affine transformation* or a *projective transformation*. An affine transformation is a special type of projective transformation, and a similarity transformation is a special type of affine transformation. These 3D transformations are defined as follows [1]:

A projective transformation in 3-space can be represented as $\mathbf{X}_\mathbf{p}=\mathbf{T}_\mathbf{p}\mathbf{X}$, where **X** is a 3D point in homogeneous coordinates, $\mathbf{X}_\mathbf{p}$ is obtained from **X** via the projective transformation, and $\mathbf{T}_\mathbf{p}$ is a 4×4 transformation matrix in the form

$$\mathbf{T}_\mathbf{p}=\begin{bmatrix}\mathbf{A} & \mathbf{t}\\ \mathbf{v}^{\mathrm{T}} & v\end{bmatrix}, \tag{2.38}$$

with **A** being a 3×3 invertible matrix, $\mathbf{t}=(t_X, t_Y, t_Z)^T$ a 3×1 translation vector, **v** a general 3×1 vector and v a scalar [1].

Similarly, an affine transformation in 3-space, $\mathbf{X_a}=\mathbf{T_a}\mathbf{X}$, is performed through a transformation matrix $\mathbf{T_a}$ of the form

$$\mathbf{T_a} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{2.39}$$

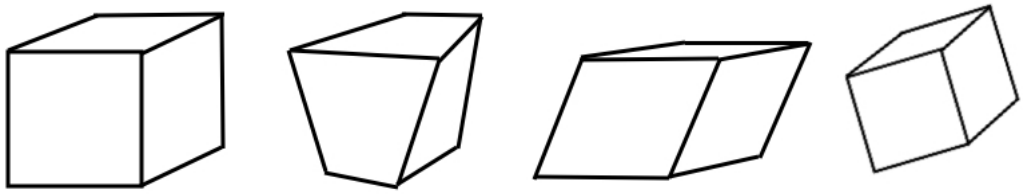
where **A** is a 3×3 invertible matrix, $\mathbf{t}=(t_X, t_Y, t_Z)^T$ is a 3×1 translation vector and $\mathbf{0}=(0, 0, 0)^T$ is the 3×1 null vector.

Finally, a similarity transformation in 3-space, $\mathbf{X_s}=\mathbf{T_s}\mathbf{X}$, is represented via a transformation matrix $\mathbf{T_s}$ of the form

$$\mathbf{T_s} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{2.40}$$

where **R** is a 3×3 rotation matrix, $\mathbf{t}=(t_X, t_Y, t_Z)^T$ is a 3×1 translation vector, and s is a scalar representing scale change.

A similarity transformation can be considered as a combination of rotation, translation and scale change. A similarity transformation preserves parallelisms, orthogonalities, distance ratios, and the image of the absolute conic (see Section 2.4.3 for more detail), whereas an affine transformation preserves parallelisms, volume ratios and centroids. A projective transformation generally does not preserve any of these. Intersection and tangency of contacting surfaces are some of the invariant properties of a projective transformation. In order to visualize their effects, typical similarity, affine and projective transformations applied to a cubic object are illustrated in Figure 2.16.



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

Figure 2.16: 3D Transformations: **(a)** The original cubic object **(b)** Projective transformation **(c)** Affine transformation **(d)** Similarity transformation applied to the object

In a reconstruction problem, if correspondence pairs are the only clues about the system, then for any reconstruction with projection matrices **P**, **P'** and a 3D point **X** satisfying

$$\mathbf{x}=\mathbf{PX}$$
$$\mathbf{x}'=\mathbf{P}'\mathbf{X},$$

it is possible to obtain an equivalently valid reconstruction

$$\mathbf{x}=(\mathbf{PH})(\mathbf{H}^{-1}\mathbf{X})$$
$$\mathbf{x}'=(\mathbf{P}'\mathbf{H})(\mathbf{H}^{-1}\mathbf{X})$$

through any 4×4 invertible matrix **H**, where **PH** and **P'H** are the projection matrices and $\mathbf{H}^{-1}\mathbf{X}$ is the 3D scene point corresponding to the new reconstruction. This is known as *projective ambiguity*, and if the calibration is unknown for both cameras, the 3D points $\{\mathbf{X}_i\}$ and cameras can be determined only up to a projective transformation, which is called *projective reconstruction* [1].

A scene can be reconstructed projectively from only correspondences even if no other information is available about cameras. Additional information about the cameras may make it possible to refine the projective reconstruction to a more specific level. If camera calibration is known, i.e. internal calibration parameters

are available, the scene can be reconstructed up to a similarity transformation [1]. This type of reconstruction is called as a *similarity reconstruction*, and is referred to as *Euclidean reconstruction* or *metric reconstruction* in the literature as well.

In this thesis, in order to reconstruct the scene, first the projective camera matrices are computed by using the fundamental matrix. Then the projective reconstruction of scene points are obtained via triangulation, where the 3D location of a scene point is determined by intersecting two rays, each of which passes through the camera center and the image of the scene point on the image plane. Finally, the obtained projective reconstruction is upgraded to achieve a metric reconstruction through a stratification matrix **H**, which is computed from the camera calibration matrix **K**.

2.4.2 Projective Reconstruction of the Scene

2.4.2.1 Determination of Projection Matrices

As epipolar geometry estimation has already been achieved in the previous stage, the first part of the projective reconstruction of the scene is the determination of suitable projective camera matrices consistent with the fundamental matrix.

It has been mentioned in the previous section that knowing the fundamental matrix determines the reconstruction up to a projective transformation. Given a fundamental matrix **F** relating the two views of a scene, the projection matrices **P** and **P'** corresponding to the two cameras must satisfy that $\mathbf{P'}^{T}\mathbf{FP}$ is skew-symmetric [1].

A particular choice for **P** and **P'** consistent with **F** is the following, which satisfies the above property:

$$\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$$
$$\mathbf{P'} = [[\mathbf{e'}]_{\times}\mathbf{F} \mid \mathbf{e'}] \tag{2.41}$$

In (2.41), **e'** is the left null vector of **F** such that $\mathbf{e'}^{T}\mathbf{F}=\mathbf{0}$, and the notation $[.]_{\times}$ denotes the cross matrix of a vector. The projective camera matrices are determined according to (2.41).

2.4.2.2 Triangulation

Once the projection matrices are determined, the next problem for the reconstruction is the computation of the 3D locations of scene points. In other words, given projection matrices **P**, **P',** and its projections on images **x**, **x',** the 3D location of the point **X** must be computed such that **PX**=**x** and **P'X**=**x'**.

The reconstruction problem, whose algebraic formulation is as stated, physically corresponds to the problem of finding the intersection of two lines, each of which passes through the respective camera center and the projection on image plane. This process is called *triangulation* and is illustrated in Figure 2.17.

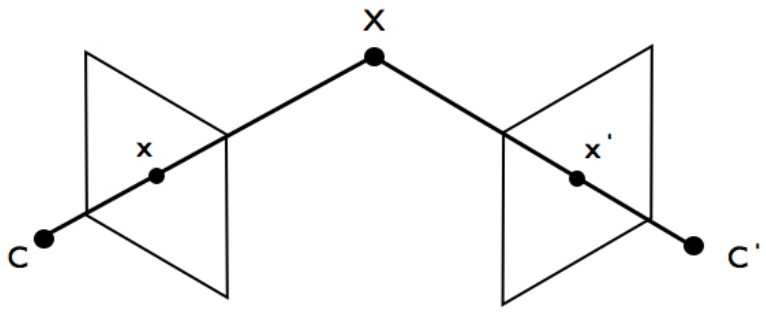


Figure 2.17: The scene point is at the intersection of the rays back-projected from the image points.

In the ideal case, where the image projection locations and projection matrices are noiseless and perfectly available, the two rays representing back-projections

intersect at a point in the 3-space, therefore, the location of the scene point can be easily determined. However, this is not the case in practical reconstruction problems, as measurement noise and computation errors are involved. If the correspondences **x** and **x'** do not satisfy the epipolar constraint (the condition $\mathbf{x'}^{T}\mathbf{F}\mathbf{x}=0$) perfectly, the rays do not meet at a point as depicted in Figure 2.18, and the optimal solution for the non-ideal reconstruction problem must be searched [1].

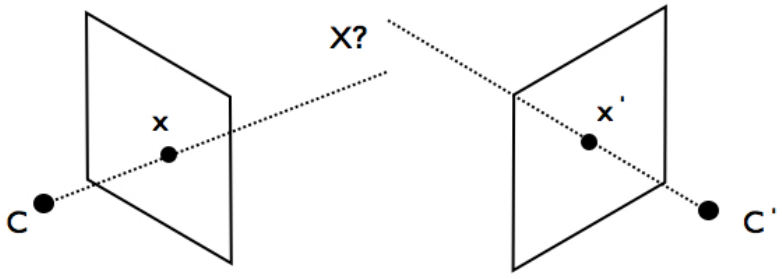


Figure 2.18: In practical problems, the back-projected rays do not intersect in space because of noise and errors.

Idealization of Correspondence Coordinates

In general, the noisy correspondence locations **x** and **x'** do not satisfy the epipolar constraint, and the correct locations of the correspondences $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$, such that $\hat{\mathbf{x}}'^{T}\mathbf{F}\hat{\mathbf{x}}=0$, are in a close neighbourhood of the noisy measurements. Therefore, triangulation of non-ideal correspondences is achievable through the solution of the optimization problem where the points $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ are searched that minimize the function

$$C(\mathbf{x},\mathbf{x}') = d(\mathbf{x},\hat{\mathbf{x}})^2 + d(\mathbf{x}',\hat{\mathbf{x}}')^2 \qquad (2.42)$$

subject to the condition $\hat{\mathbf{x}}'^{T}\mathbf{F}\hat{\mathbf{x}}=0$. The expression $d(\,.\,,\,.\,)$ in (2.42) denotes the Euclidean distance in 2D space.

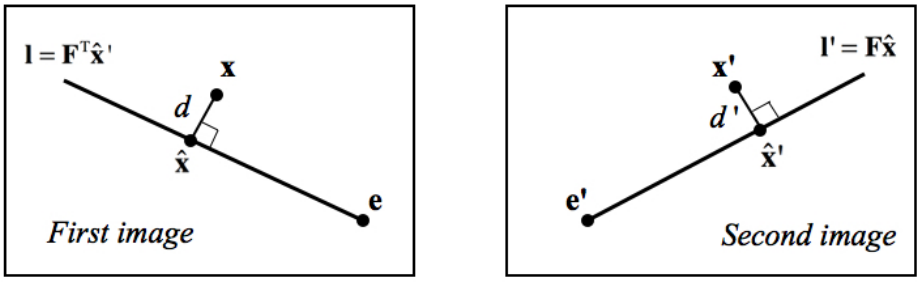


Figure 2.19: Optimization of the noisy correspondence (**x**, **x'**) to satisfy the epipolar constraint

It is possible to determine the global minimum of the cost function in (2.42) with non-iterative methods [1]. The problem is illustrated in Figure 2.19. The idealized correspondence locations $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ are sought such that $\hat{\mathbf{x}}'$ is on the epipolar line $\mathbf{l}' = \mathbf{F}\hat{\mathbf{x}}$, and $\hat{\mathbf{x}}$ is on the epipolar line $\mathbf{l} = \mathbf{F}^T\hat{\mathbf{x}}'$. However, any other correspondence pair chosen on the epipolar lines **l** and **l'** will also satisfy the epipolar constraint. Since the optimization problem stated in (2.42) requires the minimization of the distances between the idealized image points and the original measurements, the idealized correspondence locations $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ are the closest points of the epipolar lines to **x** and **x'**. Therefore, the cost function in (2.42) is equivalent to

$$C(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}, \mathbf{l})^2 + d(\mathbf{x}', \mathbf{l}')^2, \tag{2.43}$$

where **l** and **l'** are to be chosen among all possible epipolar lines.

The global minimum of the function in (2.43) can be obtained through the following procedure [1]: Firstly, the pencil of epipolar lines in the first image are parameterized by a parameter *t*, so that the epipolar lines in the first image are expressed in the form $\mathbf{l}(t)$. Then the epipolar lines in the second image, $\mathbf{l}'(t)$, are computed from **F**, which makes it possible to write the cost function in (2.43) as a function of solely *t*. Finally the value of *t* that minimizes (2.43) is computed.

In order to simplify the solution of this minimization problem, the following substitutions are made: Firstly the image points $\mathbf{x} = (u, v, 1)^T$ and $\mathbf{x'} = (u', v', 1)^T$ are moved to the origin through the transformations

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & -u \\ 0 & 1 & -v \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{T'} = \begin{bmatrix} 1 & 0 & -u' \\ 0 & 1 & -v' \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.44}$$

so that $\mathbf{x} = \mathbf{x'} = (0, 0, 1)^T$ in homogeneous coordinates. Secondly, the epipoles **e** and **e'** are normalized such that $\mathbf{e} = (e_1, e_2, e_3)^T$ and $\mathbf{e'} = (e'_1, e'_2, e'_3)^T$, where $e_1^2+e_2^2=1$ and $e'^2_1+e'^2_2=1$. Then the epipoles are placed on the x-axis as $\mathbf{e} = (1, 0, e_3)^T$ and $\mathbf{e'} = (1, 0, e'_3)^T$ via the rotations

$$\mathbf{R} = \begin{bmatrix} e_1 & e_2 & 0 \\ -e_2 & e_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R'} = \begin{bmatrix} e'_1 & e'_2 & 0 \\ -e'_2 & e'_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.45}$$

On account of these transformations, the original fundamental matrix **F** is updated as

$$\tilde{\mathbf{F}} = \mathbf{R'}(\mathbf{T'}^{-1})^T \mathbf{F} \mathbf{T}^{-1} \mathbf{R}^T, \tag{2.46}$$

which consequently has the special form

$$\tilde{\mathbf{F}} = \begin{bmatrix} e_3 e'_3 d & -e'_3 c & -e'_3 d \\ -e_3 b & a & b \\ -e_3 d & c & d \end{bmatrix}. \tag{2.47}$$

In order to parameterize the epipolar line **l**, **l**(t) is defined as the line passing through the epipole $\mathbf{e} = (1, 0, e_3)^T$ and an arbitrary point $(0, t, 1)^T$ on the y-axis.

This parameterization yields $\mathbf{l}(t) = (te_3, 1, -t)^T$, and the corresponding epipolar line $\mathbf{l}'(t)$ on the second image is obtained as $\mathbf{l}'(t) = \mathbf{F}\ (0,\ t,\ 1)^T = (-e_3'\ (ct+d),\ at+b,\ ct+d)^T$. Combining these, the cost function defined in (2.43) can be reformulated in terms of the parameter t as [1]

$$C(t) = \frac{t^2}{1+e_3^2 t^2} + \frac{(ct+d)^2}{(at+b)^2 + e'^2_3 (ct+d)^2}. \tag{2.48}$$

The minimum value of $C(t)$ is found by setting its derivative to zero and solving for t. Deriving and reorganizing $dC(t)/dt$, one can show that the local minimum occurs at the value of t which satisfies

$$\begin{aligned} g(t) &= t((at+b)^2 + e'^2_3 (ct+d)^2)^2 \\ &\quad -(ad-bc)(1+e_3^2 t^2)^2 (at+b)(ct+d) \\ &= 0 \end{aligned} \tag{2.49}$$

The function $g(t)$ in (2.49) is a 6^{th} degree polynomial, and the minimum value of $C(t)$ can be determined by computing the roots of $g(t)$ and evaluating $C(t)$ at the real roots. The asymptotic value of $C(t)$ as $t \to \infty$ must also be checked.

Once the optimum value of t is found, the optimal epipolar lines $\mathbf{l}$ and $\mathbf{l}'$ are obtained by evaluating $\mathbf{l}(t)$ and as $\mathbf{l}'(t)$ at the respective t value. Then the corrected coordinates $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ are calculated as the closest points on these lines to $\mathbf{x}$ and $\mathbf{x}'$. Applying the required back transformations, the original values of $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ are found.

Triangulation with Idealized Correspondences

Having obtained the idealized correspondence locations $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ satisfying the epipolar constraint $\hat{\mathbf{x}}'^T \mathbf{F} \hat{\mathbf{x}} = 0$, the last part of triangulation is the determination of

the location of the 3D scene point **X**. The value of **X** must be calculated which satisfies $\mathbf{PX} = \hat{\mathbf{x}}$ and $\mathbf{P}'\mathbf{X} = \hat{\mathbf{x}}'$.

The computation of **X** is easily achievable by forming a linear equation set in the form **AX=0** and solving for **X**, which is called linear triangulation. To form the matrix **A**, firstly the equation $\mathbf{PX} = \hat{\mathbf{x}}$ is reorganized to eliminate the homogeneous scale factor as

$$\hat{u}(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{1T}\mathbf{X}) = 0$$
$$\hat{v}(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{2T}\mathbf{X}) = 0$$

where $\hat{\mathbf{x}} = (\hat{u}, \hat{v}, 1)$, and $\mathbf{p}^{iT}$ denotes the i^{th} row of **P**. Writing the counterparts of these equations for the other image point $\hat{\mathbf{x}}'$ as well, one gets the equation system **AX=0**, where **A** is

$$\mathbf{A} = \begin{bmatrix} \hat{u}\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ \hat{v}\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ \hat{u}'\mathbf{p}'^{3T} - \mathbf{p}'^{1T} \\ \hat{v}'\mathbf{p}'^{3T} - \mathbf{p}'^{2T} \end{bmatrix}. \tag{2.50}$$

Once the **A** matrix is formed, the 3D location of the scene point **X** can be determined by solving the system **AX=0**.

An overview of the aforementioned triangulation algorithm is the following:

Algorithm Overview: Optimal Triangulation

Input: The projection matrices **P**, **P'** and the fundamental matrix **F** corresponding to two views, and the images **x**, **x'** of a 3D point **X** on these two views
Output: The location of **X** in 3-space

1. Translate the points **x**, **x'** to origin via the transformations given in (2.44).

2. Normalize and place the epipoles **e**, **e'** on the x-axis via the rotations given in (2.45).

3. Update **F** as in (2.46).

4. Construct the polynomial in (2.49) by determining the coefficients from (2.47)

5. Find the roots of the polynomial and determine the optimum value of t.

6. Obtain the optimal epipolar lines **l**, **l'**; and determine the corrected coordinates $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ by finding the closest points on these lines to **x** and **x'.**

7. Transform the coordinates $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ back for the original problem by replacing $\hat{\mathbf{x}}$ with $\mathbf{T}^{-1}\mathbf{R}^{T}\hat{\mathbf{x}}$, and $\hat{\mathbf{x}}'$ with $\mathbf{T}'^{-1}\mathbf{R}'^{T}\hat{\mathbf{x}}'$.

8. Construct the matrix **A** in (2.50) and calculate the point **X** as the null vector of **A**.

2.4.3 Upgrading the Projective Reconstruction to Metric

Once the projective camera matrices are constructed and 3D scene points are determined up to a projective transformation as explained in Section 2.4.2, the next step is upgrading the projective reconstruction to metric using the camera calibration information. Given a scene point $\mathbf{X_p}$ reconstructed projectively, and the images $\mathbf{x}$, $\mathbf{x'}$ of $\mathbf{X_p}$, upgrading the projective reconstruction to metric is possible by finding a 4×4 rectifying homography $\mathbf{H}$ in 3-space such that

$$\begin{aligned} \mathbf{x} &= \mathbf{P_p X_p} = (\mathbf{P_p H})(\mathbf{H}^{-1}\mathbf{X_p}) = \mathbf{P_m X_m} \\ \mathbf{x'} &= \mathbf{P'_p X_p} = (\mathbf{P'_p H})(\mathbf{H}^{-1}\mathbf{X_p}) = \mathbf{P'_m X_m} \end{aligned} \tag{2.51}$$

where $\mathbf{P_m} = \mathbf{P_p H}$ and $\mathbf{P'_m} = \mathbf{P'_p H}$ are the stratified projection matrices conforming to the pinhole camera model, and $\mathbf{X_m} = \mathbf{H}^{-1}\mathbf{X_p}$ is the metric reconstruction for the 3D scene point of interest, which determines its true location up to a similarity transformation.

A solution for the determination of $\mathbf{H}$ is presented in [1] as follows: Given a projective reconstruction $\{\mathbf{P}^i, \mathbf{X}_j\}$, where i and j denote the indices of the cameras and reconstructed points, such that the first camera matrix is $\mathbf{P}^1$=[**I** | **0**], it is possible to obtain a metric reconstruction $\{\mathbf{P}^i\mathbf{H}, \mathbf{H}^{-1}\mathbf{X}_j\}$ through the following rectification matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ -\mathbf{p}^{\mathrm{T}}\mathbf{K} & 1 \end{bmatrix}, \tag{2.52}$$

where $\mathbf{K}$ is the calibration matrix of the first camera, and the coordinates of the plane at infinity in the projective reconstruction are $\pi_\infty = (\mathbf{p}^{\mathrm{T}}, 1)^{\mathrm{T}}$.

Taking these relations into account, the problem of stratifying the projective reconstruction is reduced to the problem of determining the plane at infinity. The plane at infinity is defined as the plane in 3-space with the canonical position $\pi_\infty = (0, 0, 0, 1)^T$ [1]. The criticality of π_∞ arises from the fact that it is fixed under any affine transformation, but is moved by a projective transformation. It is the property of invariance under affine transformations that makes the plane at infinity beneficial in some reconstruction problems.

It is possible to determine the coordinates of π_∞ for a projective reconstruction by making use of the relation between the camera calibration matrix and the image of the absolute conic [1]. The absolute conic, denoted by Ω_∞, is a conic on π_∞ such that any point $\mathbf{X}=(X_1, X_2, X_3, X_4)^T$ on Ω_∞ satisfies:

$$\begin{aligned} X_1^2 + X_2^2 + X_3^2 &= 0 \\ X_4 &= 0 \end{aligned} \tag{2.53}$$

As (2.53) reveals, the absolute conic is a conic that consists of purely imaginary points on the plane at infinity. An important property of Ω_∞ is that the image of the absolute conic, denoted by ω, is independent of the position of the camera, and depends only on the camera calibration matrix **K**. The relation between ω and **K** is given by [1]

$$\omega = (\mathbf{K}\mathbf{K}^T)^{-1}. \tag{2.54}$$

From the image of the absolute conic, the plane at infinity can be determined with the following approach: It has been shown that [18], given two projection matrices of form **P**=[**I** | **0**] and **P'**=[**A** | **e'**], the fundamental matrix **F** relating the two views, and the images of a conic **C**, **C'** in two views, where **e** and **e'** denote the epipoles in the first and second images, it is possible to determine the plane π

on which the conic resides, up to a two-fold ambiguity. The plane π is in the form $\pi^T = (\mathbf{a}^T, 1)^T$, where $\mathbf{a}(\mu)$ is given by

$$\mathbf{a}(\mu) = -\left(\mu \mathbf{Ce} + \mathbf{A}^T \mathbf{C'e'}\right) / \left(\mathbf{e'}^T \mathbf{C'e'}\right), \tag{2.55}$$

and the two possible values for μ are determined from

$$\mu^2 \left[(\mathbf{e}^T \mathbf{Ce})\mathbf{C} - (\mathbf{Ce})(\mathbf{Ce})^T \right] = \mathbf{A}^T \left[(\mathbf{e'}^T \mathbf{C'e'})\mathbf{C'} - (\mathbf{C'e'})(\mathbf{C'e'})^T \right] \mathbf{A} \tag{2.56}$$

In order to find the plane at infinity, the conic images **C** and **C'** in equations (2.55) and (2.56) are chosen as the images of the absolute conic. Setting $\mathbf{C} = \mathbf{C'} = \omega = (\mathbf{KK}^T)^{-1}$ in (2.56), one obtains two values for μ from (2.56). Each of the two values of μ in (2.56) defines a different solution for π_∞, which in turn yields two different metric reconstructions for the same projective reconstruction, where only one of the metric reconstructions will be the valid one. This ambiguity is associated with the twisted pair ambiguity [19], which states that for any valid metric reconstruction consisting of camera matrices and scene points, it is possible to define an alternative metric reconstruction that yields the same projections on the image planes.

The procedure for refining a projective reconstruction to obtain a metric reconstruction is summarized as follows:

Algorithm Overview: Upgrading a Projective Reconstruction to Metric

Input: Two projective camera matrices of the form $\mathbf{P}_p = [\mathbf{I}|\mathbf{0}]$, $\mathbf{P}_p' = [\mathbf{A}|\mathbf{e'}]$, scene points $\{\mathbf{X}_p\}$ reconstructed projectively, the fundamental matrix **F** relating the two views, and the camera calibration matrix **K**

Output: Stratified metric reconstruction $\mathbf{P}_m$, $\mathbf{P}_m'$, $\{\mathbf{X}_m\}$ determined up to a two-fold ambiguity

1. Compute the image of the absolute conic, $\omega=(\mathbf{K}\mathbf{K}^T)^{-1}$.

2. Set $\mathbf{C}=\mathbf{C}'=\omega$ in (2.56) and calculate the two values of μ.

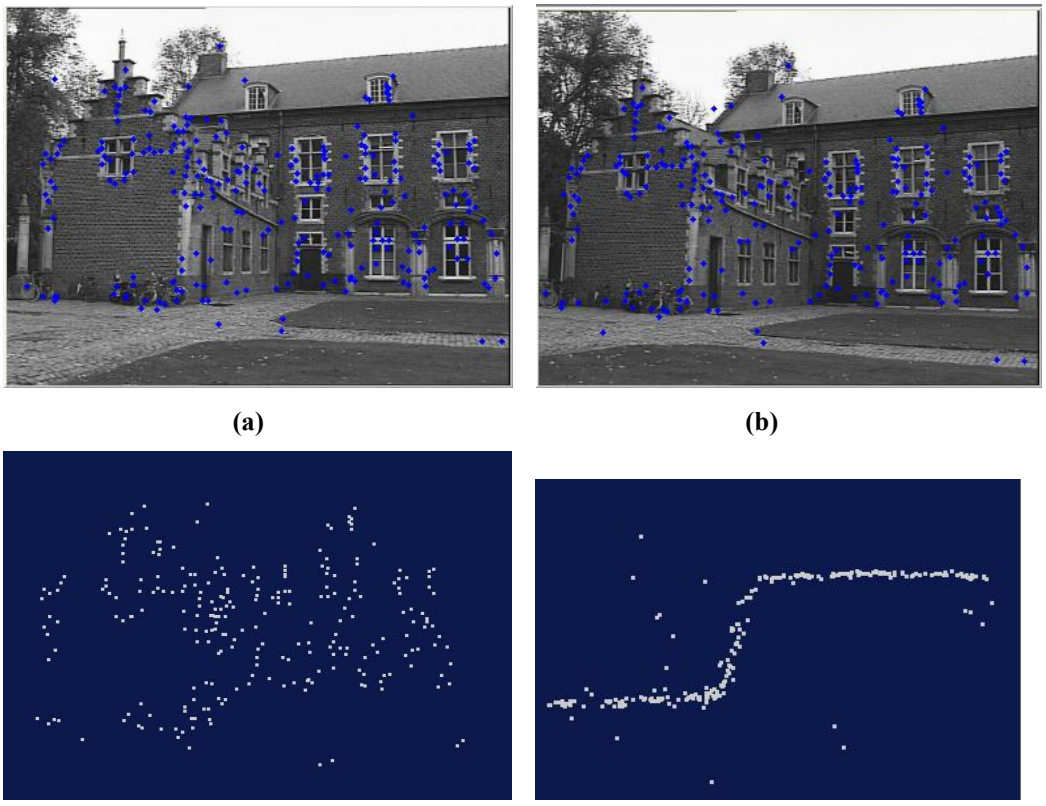
3. For each value of μ, determine the corresponding plane at infinity using (2.55).

4. Compute the rectifying homography $\mathbf{H}$ for each plane at infinity from (2.52).

5. Obtain the metric reconstruction $\mathbf{P}_m=\mathbf{P}_p\mathbf{H}$, $\mathbf{P}_m'=\mathbf{P}_p'\mathbf{H}$, $\mathbf{X}_m=\mathbf{H}^{-1}\mathbf{X}_p$ for both values of $\mathbf{H}$.

2.4.4 Experimental Results

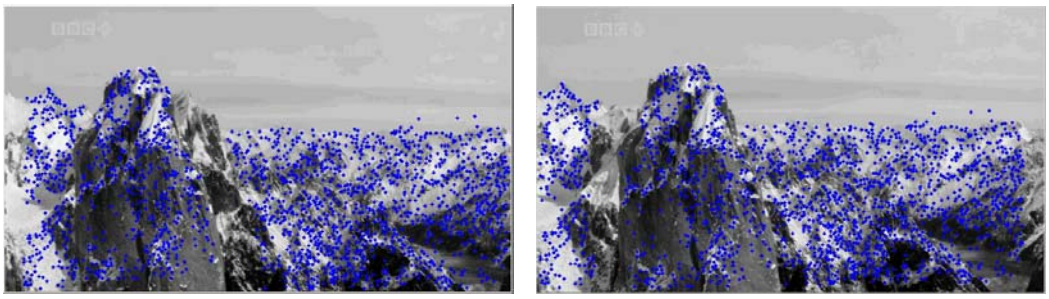
The studied reconstruction algorithm, which first performs projective reconstruction using the fundamental matrix and correspondences, and then upgrades the projective reconstruction to metric, is applied to two images from the sequences *Castle*, *Planet Earth* and *Krimm*. In all experiments, feature detection is achieved via SIFT, and the epipolar geometry is estimated using the 7-point algorithm and RANSAC. The results are presented in Figures 2.20, 2.21 and 2.22. In each of the figures, (a) and (b) show the features detected (and matched) on the first and second images respectively, whereas in (c) the front view and in (d) the top view of the reconstructed scene are given.



(a) **(b)**

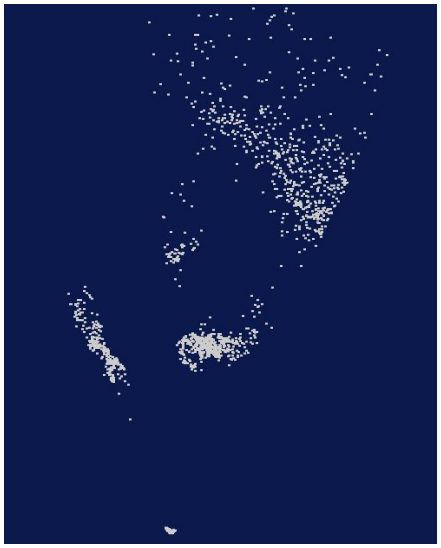
(c) **(d)**

Figure 2.20: Two-view reconstruction results for *Castle*



(a) **(b)**

(c)



(d)

Figure 2.21: Two-view reconstruction results for *Planet Earth*

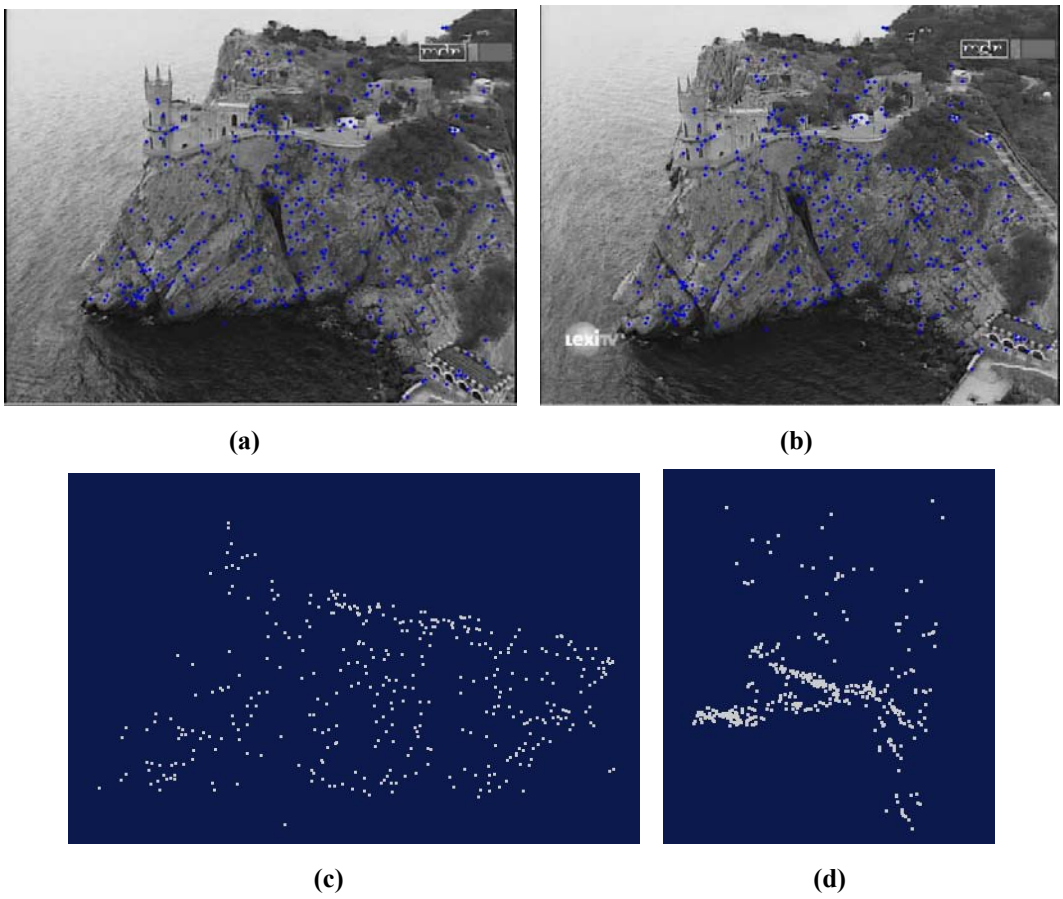


Figure 2.22: Two-view reconstruction results for *Krimm*

2.5 Experimental Comparison of Competing Algorithms Used in Sparse Reconstruction

In this section, the results of some experiments comparing the performances of opponent algorithms to be used in sparse reconstruction are presented. In Section 2.5.1, SIFT is evaluated by comparing its performance to those of Harris corner detector and KLT tracker from several aspects. In Section 2.5.2, the performances of RANSAC and PROSAC in epipolar geometry estimation are compared, and in Section 2.5.3, the comparison of the 7-point and 8-point algorithms is made.

2.5.1 Experimental Comparison of SIFT, Harris Corner Detector and KLT Tracker

In order to assess the value of SIFT as a feature detector, experiments have been conducted where SIFT is compared to two other well- known algorithms in the literature, namely the Harris corner detector [3] and the Kanade-Lukas-Tomasi (KLT) [20] tracker.

KLT tracker is a tool for obtaining trajectories of feature points in an image sequence. The tracker initiates the trajectories by running Harris corner detector in the first frame of the sequence, and then estimates the displacements of these features in the proceeding frames with an optical-flow based approach, hence, obtains trajectories of features throughout the sequence. In the tests, the optical-flow based tracker was used, which integrates a Kalman filter into the basic KLT tracker to improve its performance [28].

In the experiments, the performances of three algorithms were evaluated by the quality of the matches they provide. The algorithms were applied to video data to obtain matches in the following manner:

- SIFT and Harris corner detector: Two frames with 15-frame distance in between were chosen from the video sequence, and features were detected with the tested algorithms. The features detected by SIFT were matched via the feature matching tool of SIFT, whereas in the case of Harris corner detector, feature detection was achieved by searching the image intensity correlations around feature locations [9]. The outputs of the algorithms, which are lists of correspondences, were directly used in the test.
- KLT tracker: The tracker was run on the whole sequence. In order to obtain correspondence lists belonging to the selected two frames, the related part of the trajectories corresponding to the selected frames were extracted.

The tests provide a direct comparison between SIFT and Harris corner detector (combined with the intensity correlation based feature matching algorithm), as both are feature detection and matching tools operating on the same set of inputs, namely, two images extracted from the video. However, it must be remembered that, unlike SIFT and Harris corner detector, KLT tracker is a feature tracking algorithm which runs on a set of consecutive video frames. Hence, as far as the KLT tracker is concerned, the conducted experiments provide a comparison only between the two approaches for estimating the geometry between two video frames, one of which utilizes features and matches detected only on the selected frames, where the other achieves this via tracking the features benefiting from the whole set of frames between the selected two.

During the experiments, the qualities of match lists were assessed by testing their amenability to epipolar geometry estimation. For this purpose, the correspondence lists were processed by RANSAC to estimate the fundamental matrix. The successes of the tested algorithms were compared with respect to the following criteria:

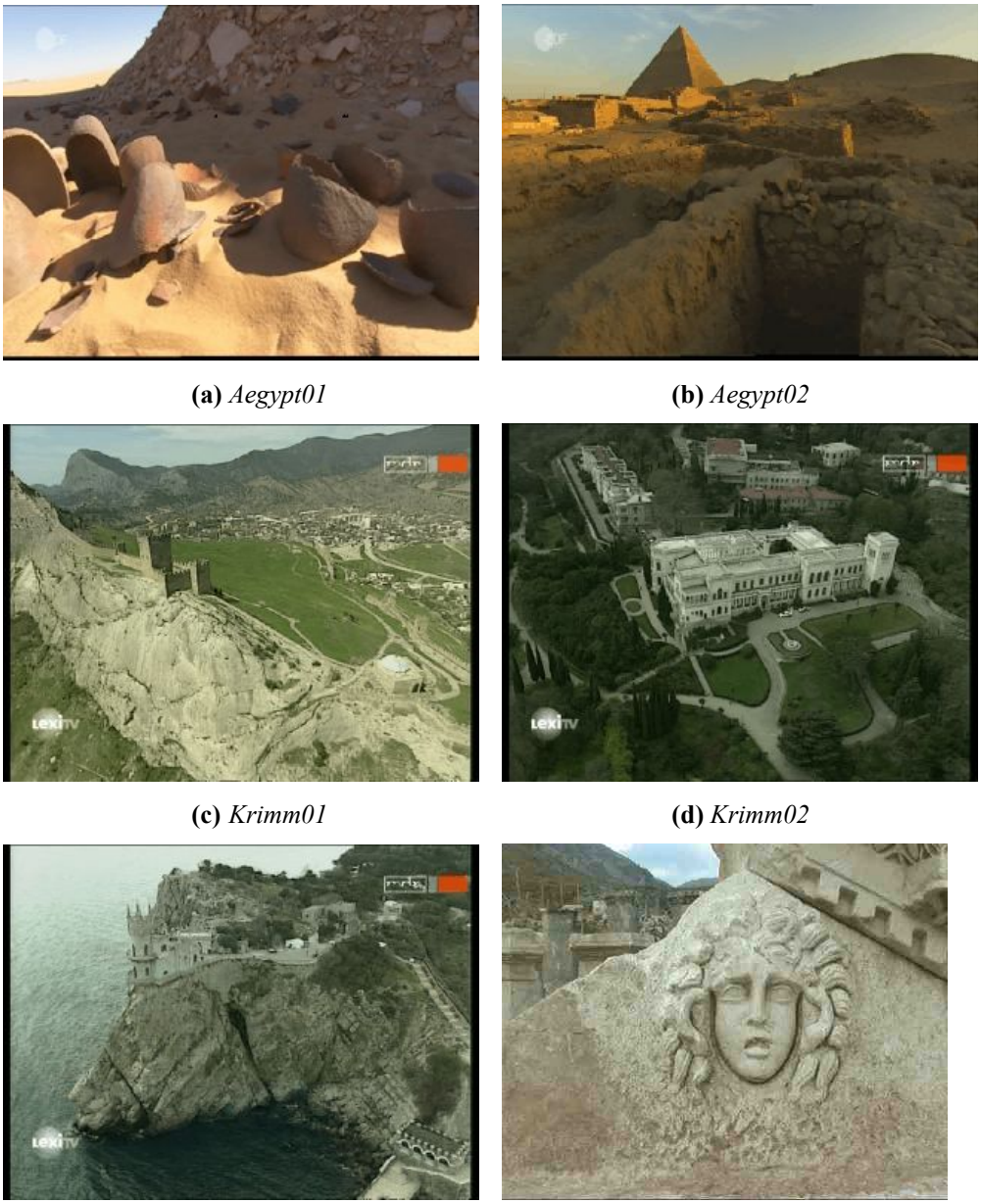
- Inlier / Total match number ratio: The capability of the feature detection algorithm affects the number of correct matches; therefore, the inlier/total match number ratio is a measure of the quality of the feature detection method.
- Sampson error per inlier: Since Sampson error is a quantity that measures the consistency of a match with the epipolar geometry, a small average Sampson error indicates a high accuracy in the localization of the matching points.
- Iteration number of RANSAC: RANSAC is an adaptive algorithm that determines its termination time depending on the success of the samples it draws. Therefore, the number of iterations should get smaller as the ratio of correct matches among the whole set increases.
- Total variance of fundamental matrix entries: In order to make robust epipolar geometry estimation, the consistency in the determination fundamental matrix must be assured. Therefore the total variance of fundamental matrix entries can be used to assess the robustness of the epipolar geometry determined from the matches.

For comparing the performances of the algorithms with respect to these criteria, a series of experiments have been conducted with several video data. In each experiment, a set of correspondences was obtained by the tested algorithm, and RANSAC was run on this set of correspondences 100 times. The inlier/total ratio, Sampson error per inlier and the iteration number was taken as the average of these 100 trials, and the total variance of the fundamental matrix was estimated using the observations of matrix entries during the trials. The upper bound for the number of iterations for RANSAC was set to 2000.

In Table 2.1, a summary of the experiment results is presented. Blue and red colours indicate the superiority of SIFT and KLT tracker respectively. A sample frame from each video data is given in Figure 2.23.

Table 2.1: Experimental comparison of SIFT, Harris corner detector and KLT tracker: The columns from left to right indicate: Video data, Algorithm, Total number of correspondences, Number of inliers, Inlier/Total match number ratio, Sampson error per inlier, Number of iterations, Total variance of fundamental matrix entries

Data	Algorithm	# Total cor.	# Inliers	I/T Ratio	S. Error	# Iter.	F var.
	SIFT	630	547.270	0.869	0.066	22.690	0.778
Aegypt01	*Tracker*	1185	1147.740	0.969	0.039	8.910	0.750
	Harris	767	376.680	0.491	0.089	1862.520	0.995
	SIFT	179	157.250	0.878	0.061	20.880	0.750
Aegypt02	*Tracker*	549	525.900	0.958	0.037	10.540	0.960
	Harris	211	111.810	0.530	0.086	766.430	0.989
	SIFT	630	547.320	0.869	0.080	38.100	0.979
Krimm01	*Tracker*	5363	3719.610	0.694	0.983	194.130	0.959
	Harris	1258	466.560	0.371	0.104	2000.000	0.990
	SIFT	332	243.210	0.733	0.084	118.670	0.957
Krimm02	*Tracker*	4519	2500.210	0.553	0.109	983.920	0.939
	Harris	1203	299.350	0.249	0.102	2000.000	0.999
	SIFT	361	268.840	0.745	0.091	83.080	0.806
Krimm03	*Tracker*	4639	2964.150	0.639	0.105	332.260	0.994
	Harris	1258	221.290	0.176	0.111	2000.000	0.956
	SIFT	454	411.770	0.907	0.067	20.530	0.952
Medusa	*Tracker*	4780	2647.770	0.554	0.112	719.240	0.998
	Harris	632	260.600	0.412	0.952	2000.000	0.998



(a) *Aegypt01* **(b)** *Aegypt02*

(c) *Krimm01* **(d)** *Krimm02*

(e) *Krimm03* **(f)** *Medusa*

Figure 2.23: Sample frames from the video data used in the experiments

The results presented in Table 2.1 lead to the following observations: In all experiments, SIFT yields more inliers, a higher inlier/total match number ratio, a smaller average Sampson error and a smaller fundamental matrix total variance against Harris corner detector. Therefore the performance of SIFT is found out to be significantly better than that of Harris corner detector. However, it must also be noted that computational cost of SIFT is much higher than that of Harris corner detector.

On the other hand, when the successes of SIFT and the tracker are compared, it is observed that none of the algorithms has an immediate superiority to the other. The tracker has produced a greater number of total correspondences and inliers than SIFT, but the inlier/total match number ratio is most of the time higher in SIFT. The average Sampson errors are close to each other, and in most experiments the iteration number of SIFT is smaller than that of tracker in a manner consistent with the inlier/total match number ratio. The total variances of the fundamental matrices are close to each other most of the time, and sometimes smaller in SIFT. Therefore an overall comment on the performances of SIFT and the tracker may be that tracker is better at providing a large number of inlier matches, and SIFT usually produces a higher inlier/total match number ratio.

2.5.2 Experimental Comparison of RANSAC and PROSAC

Being a well-known and frequently used method in the literature, RANSAC is taken as a reference to assess the performance of PROSAC. Therefore, a series of experiments have been made that compare RANSAC and PROSAC.

In all experiments, the same input set of matches were given to RANSAC and PROSAC. The input sets of correspondences were produced by running SIFT on the images. The quality of the correspondence set, i.e. the ratio of correct matches among the whole set of matches, was adjusted by changing the matching

threshold of SIFT, i.e. the threshold for the distance ratio, which also enabled to control the inlier/total ratio during the experiments. The behaviours of the algorithms were examined and compared with respect to the following criteria:

- Inlier/Total match number ratio at various values of SIFT matching threshold
- The number of iterations at various values of inlier/total match number ratio
- Sampson error per inlier at various values of inlier/total match number ratio
- The total variance of fundamental matrix entries

In each experiment, both algorithms were run 100 times. The inlier/total ratio, average Sampson error and the number of iterations were taken as the average of all trials, and the total variance of fundamental matrix entries was estimated from the models computed during these trials. Experiments were repeated for several video data, some sample frames belonging to which are given in Figure 2.23. In Figure 2.24, the inlier/total ratios obtained from PROSAC and RANSAC are presented for various values of SIFT matching threshold. Figures 2.25, 2.26 and 2.27 show respectively the number of iterations, Sampson error per inlier and the total variance of fundamental matrix entries with respect to the inlier/total ratio.

The plots presented in Figure 2.24 reveal that the inlier/total match number ratios yielded by RANSAC and PROSAC are close to each other for the same input sets of matches. At smaller values of SIFT matching threshold, the inlier/total match number ratio is in favour of RANSAC, however, as the matching threshold of SIFT increases, the difference between the algorithms is reduced, and finally PROSAC generally becomes slightly more advantageous at the highest matching threshold.

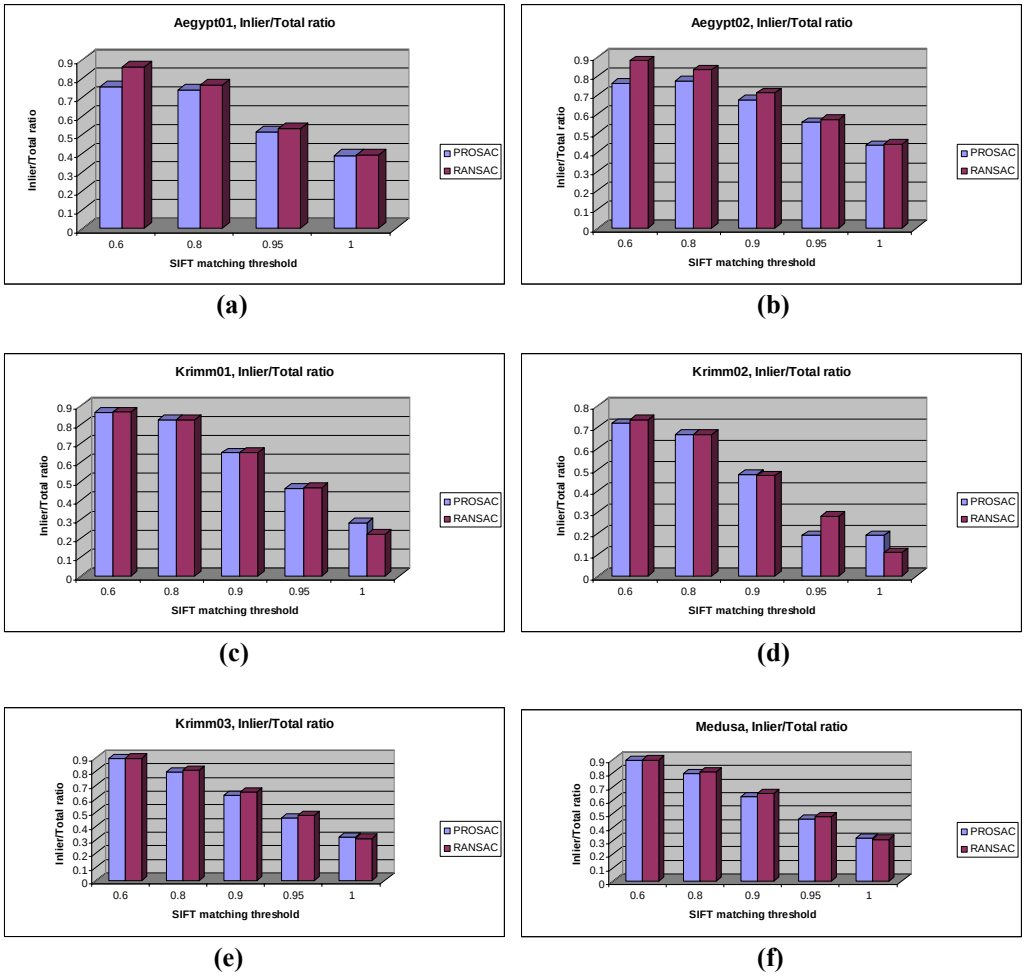


Figure 2.24: Inlier/Total match number ratios obtained at various SIFT matching thresholds for the data: **(a)** *Aegypt01* **(b)** *Aegypt02* **(c)** *Krimm01* **(d)** *Krimm02* **(e)** *Krimm03* **(f)** *Medusa*, with RANSAC and PROSAC

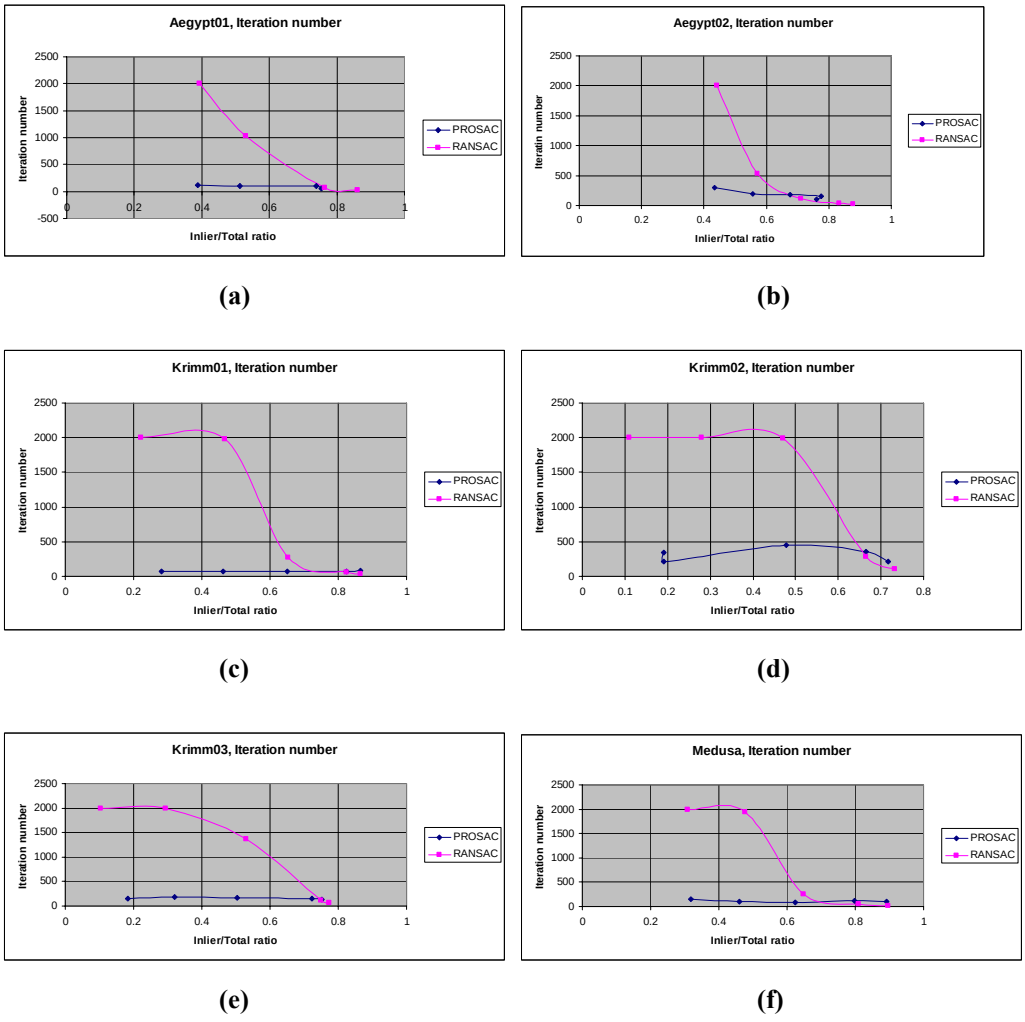


Figure 2.25: Iteration numbers obtained at various values of inlier/total match number ratio for the data: **(a)** *Aegypt01* **(b)** *Aegypt02* **(c)** *Krimm01* **(d)** *Krimm02* **(e)** *Krimm03* **(f)** *Medusa*, with RANSAC and PROSAC

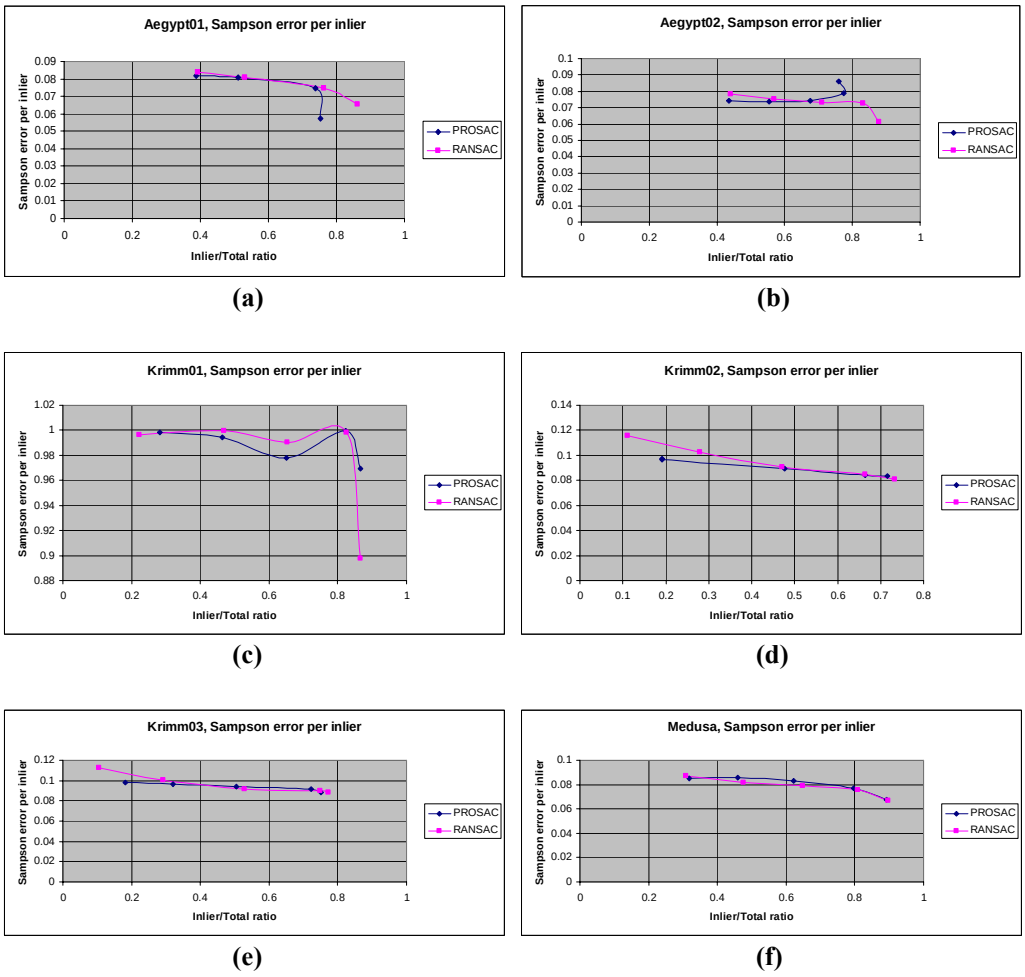


Figure 2.26: Average Sampson errors obtained at various values of inlier/total match number ratio for the data: (**a)** *Aegypt01* **(b)** *Aegypt02* **(c)** *Krimm01* **(d)** *Krimm02* **(e)** *Krimm03* **(f)** *Medusa*, with RANSAC and PROSAC

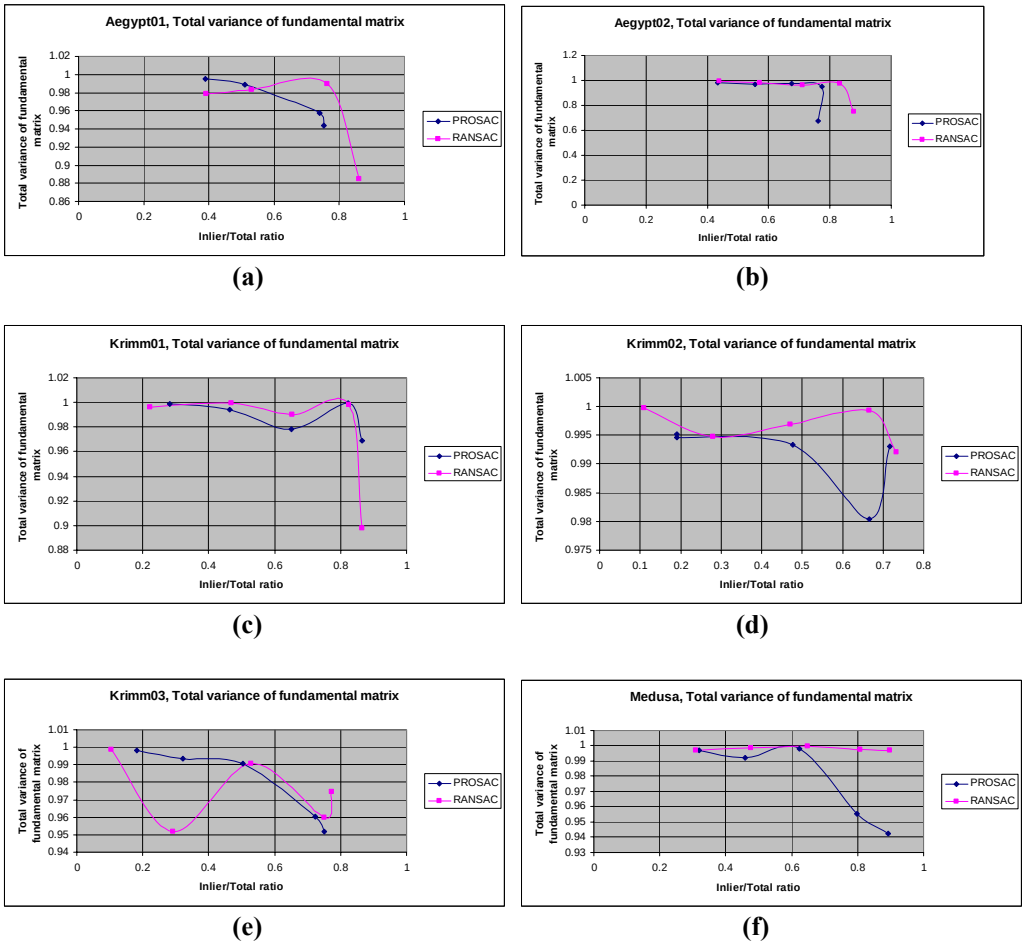


Figure 2.27: Sum of the variances of fundamental matrix entries at various values of inlier/total match number ratio for the data: (**a**) *Aegypt01* **(b)** *Aegypt02* **(c)** *Krimm01* **(d)** *Krimm02* **(e)** *Krimm03* **(f)** *Medusa*, with RANSAC and PROSAC

The plots of Figure 2.25 demonstrate the most prominent advantage of PROSAC over RANSAC. It is seen that PROSAC terminates at a number of iterations much less than RANSAC for small values of the inlier/total ratio. However, at high values of the inlier/total ratio, the iteration number of RANSAC is below that of PROSAC, which is consistent with the reasoning that the randomly selecting strategy of RANSAC has more chances when the quality of the input set of correspondences increases.

In Figures 2.26 and 2.27 some plots are given which compare RANSAC and PROSAC with respect to average Sampson error and total variance of the fundamental matrix. These plots suggest that none of the algorithms is superior to the other as far as these criteria are concerned.

An overall conclusion drawn from these experiments can be stated as follows: The performances of PROSAC and RANSAC are in general similar, however, when the ratio of inliers to the whole set of correspondences is small, i.e. mismatches are likely to be observed, PROSAC terminates at a shorter time than RANSAC, which makes it more favourable for such occasions.

2.5.3 Experimental Comparison of 7-point and 8-point Algorithms

The final part of the comparative tests is the comparison of the performances of the 7-point and 8-point algorithms. In this setup of experiments, an input set of matches was obtained with SIFT, and the sampling strategy in epipolar geometry estimation was RANSAC, where the 7-point and 8-point algorithms were tested in the estimation of the F-matrix. The purpose of the experiments was to observe the effect of these algorithms on the estimated model, therefore, as in the previous experiments, the criteria used in the evaluation of the algorithms were:

- Inlier/Total match number ratio at various values of SIFT matching threshold
- The number of iterations at various values of inlier/total match number ratio
- Sampson error per inlier at various values of inlier/total match number ratio

The results were again obtained as the average of 100 trials and the experiments were conducted on the same data as in the previous sections. During the iterations of RANSAC, model estimation was done with the algorithm under search, i.e. with either 7-point or 8-point algorithm. However, the final refinement of the model using all inliers was always achieved with the 8-point algorithm, as the seven point algorithm accepts only seven correspondences.

In Figure 2.28, the plots showing the inlier/total match number ratios at several SIFT matching thresholds are presented, whereas Figures 2.29 and 2.30 show respectively the variations of the average Sampson error and the number of iterations with respect to the inlier/total match number ratios.

The results of these experiments can be summarized as follows: The plots in Figure 2.28 show that the inlier/total match number ratios are in general very close for the 7-point and 8-point algorithms. However, the 7-point algorithm tends to give a slightly higher ratio of inliers for some data. On the other hand, Figure 2.29 reveals that for all sequences, the average number of iterations made by the 7-point algorithm is less than that of the 8-point algorithm. Finally, the plots in Figure 2.30 show that the two algorithms perform similar as far as the average Sampson error is concerned.

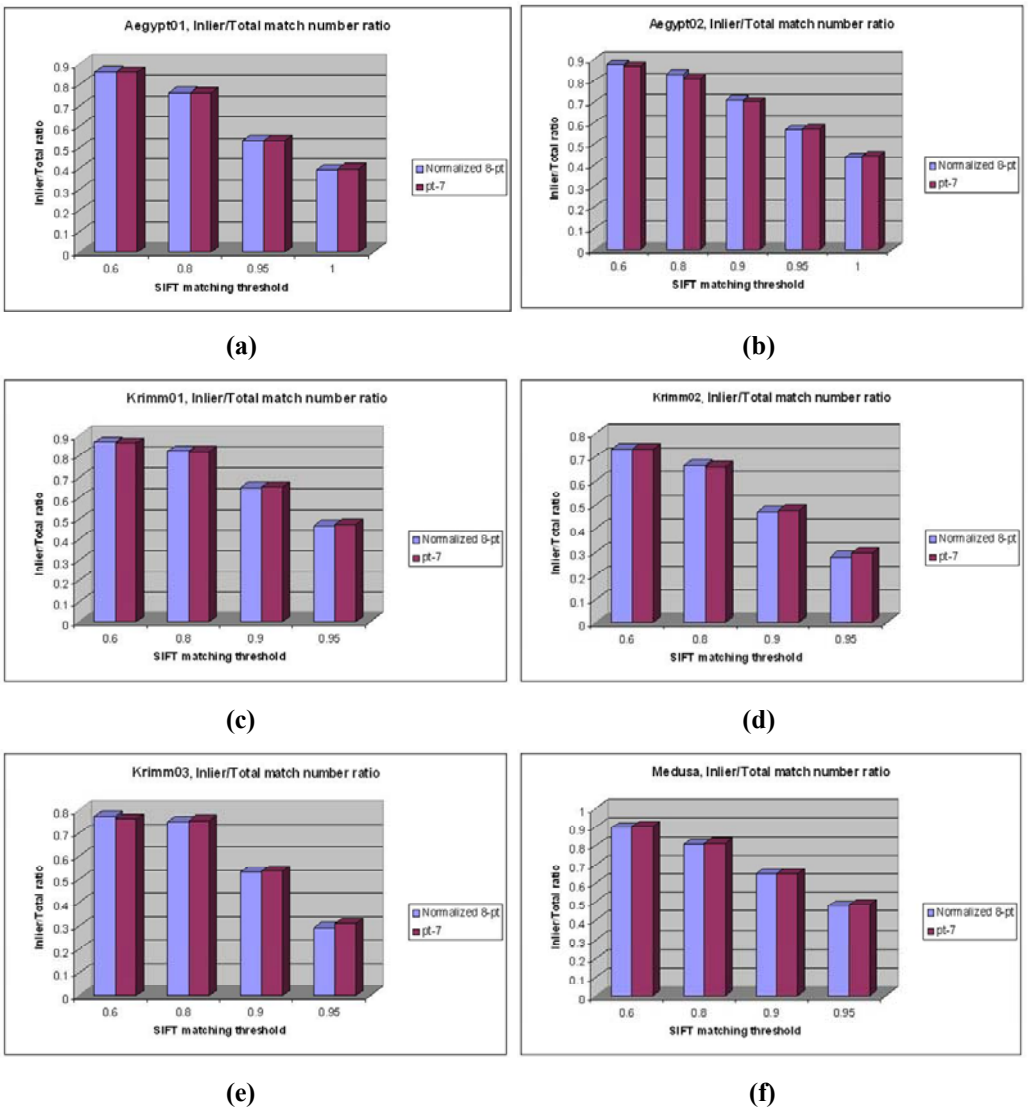


Figure 2.28: Inlier/Total match number ratios obtained at various SIFT matching thresholds for the data: (**a)** *Aegypt01* **(b)** *Aegypt02* **(c)** *Krimm01* **(d)** *Krimm02* **(e)** *Krimm03* **(f)** *Medusa*, with 7-point and 8-point algorithms

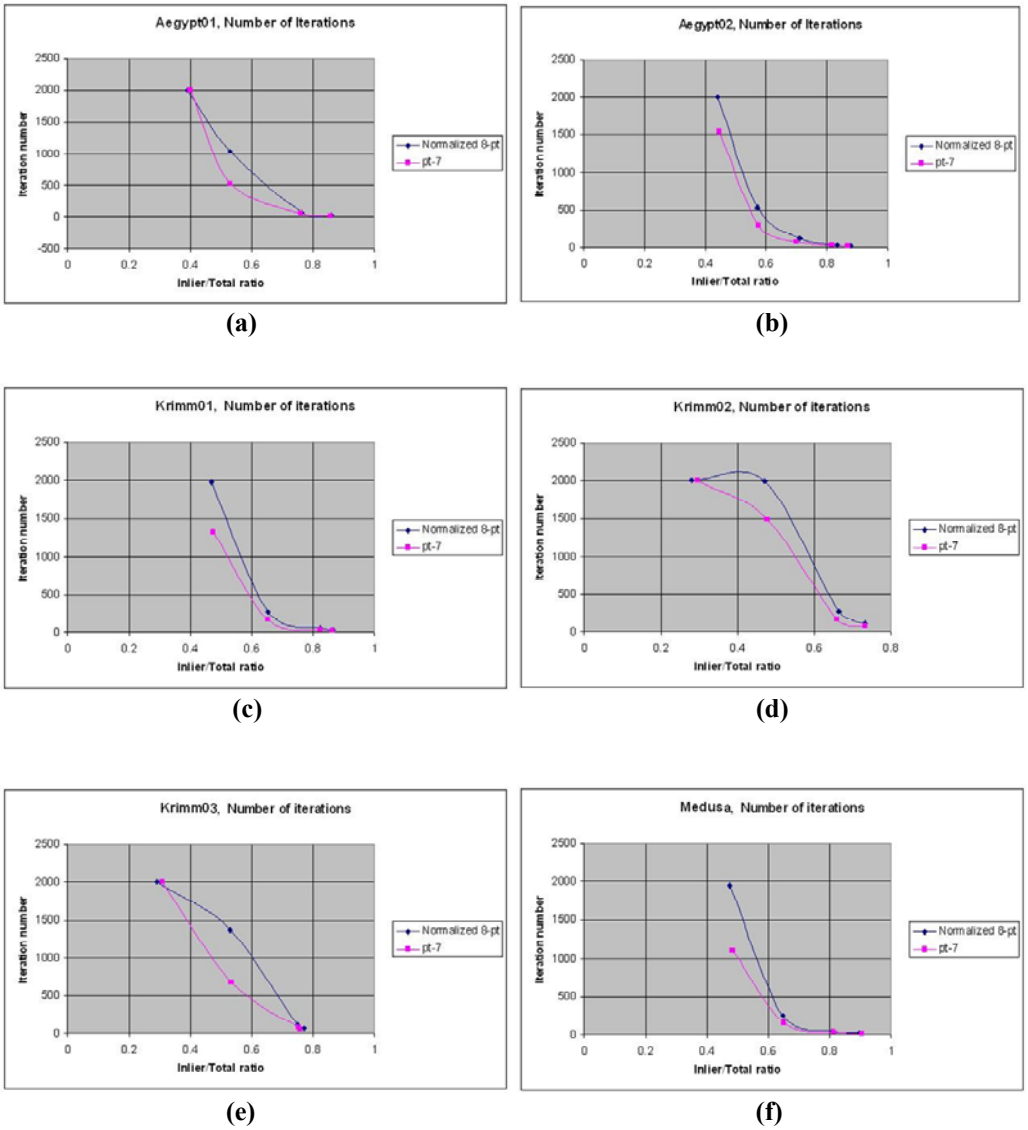


Figure 2.29: Iteration numbers obtained at various values of inlier/total match number ratio for the data: (**a)** *Aegypt01* **(b)** *Aegypt02* **(c)** *Krimm01* **(d)** *Krimm02* **(e)** *Krimm03* **(f)** *Medusa*, with 7-point and 8-point algorithms

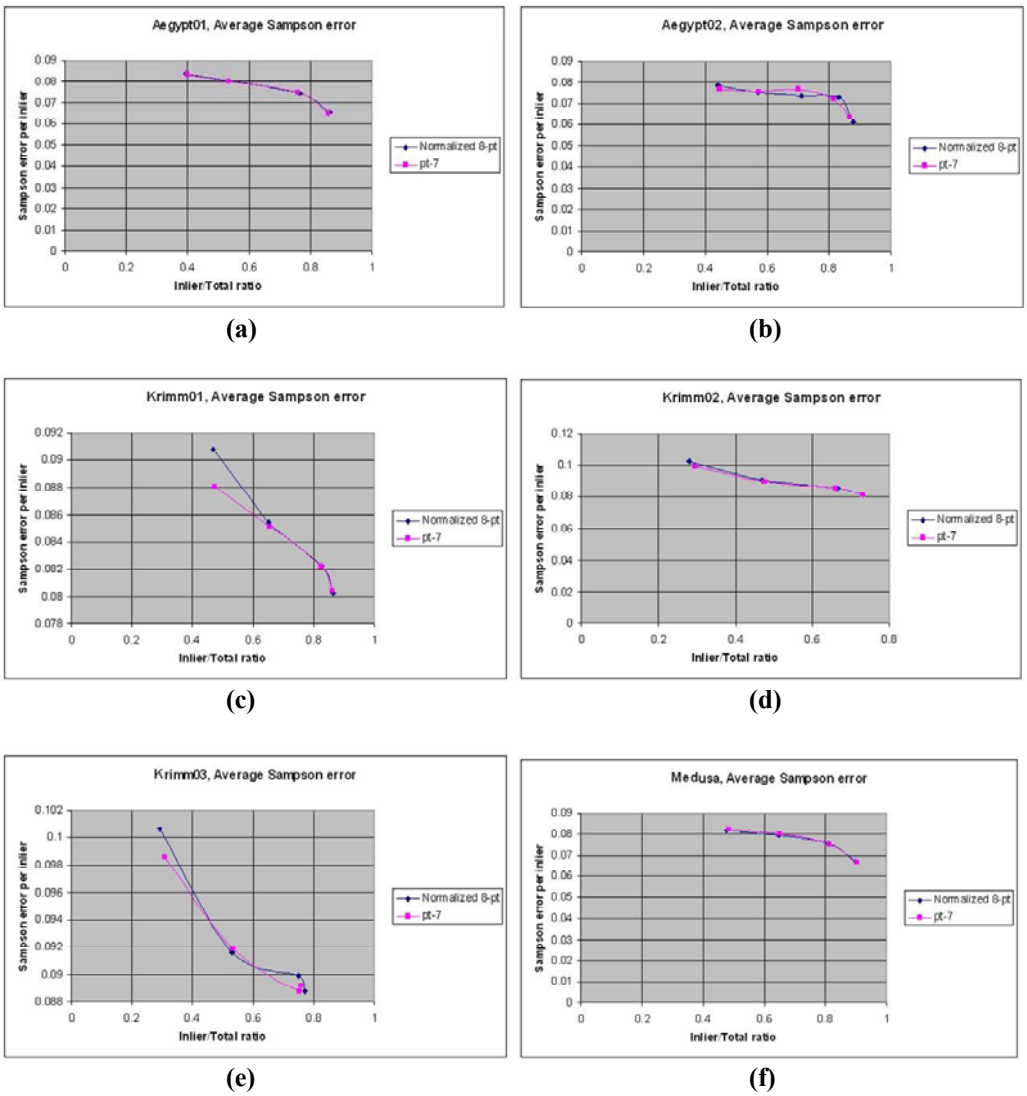


Figure 2.30: Average Sampson errors obtained at various values of inlier/total match number ratio for the data: (**a)** *Aegypt01* **(b)** *Aegypt02* **(c)** *Krimm01* **(d)** *Krimm02* **(e)** *Krimm03* **(f)** *Medusa*, with 7-point and 8-point algorithms

These comparative tests suggest that the performances of 7-point and 8-point algorithms are similar, whereas the 7-point algorithm may give slightly better results. This observation is reasonable, considering the fact that the fundamental matrix is implicitly required to be rank-2 in the 7-point algorithm, whereas the original output of the 8-point algorithm is not necessarily rank-2, and is artificially converted to the closest rank-2 matrix, which might degrade the quality of the matrix.

CHAPTER 3

3D SPARSE RECONSTRUCTION OF A SCENE FROM MULTIPLE VIEWS

In this chapter, sparse reconstruction of a scene from multiple views is examined. The problem of estimating the structure and cameras from two views is addressed in Chapter 2, and multiple view reconstruction is achieved based on the method used in two-view reconstruction, with an approach similar to the one presented in [2]. In order to obtain the sparse 3D model of a scene from an image sequence, first an initial two-view reconstruction is performed from the first two images, and then this reconstruction is extended by estimating the camera poses for each additional view, and including the 3D points obtained from these views in the reconstruction.

In addition to obtaining the sparse model of the scene, the problem of eliminating the erroneous points from the reconstruction is also studied in this chapter. Two methods are proposed for the elimination of outliers. After the outliers are discarded, the camera matrices and the locations of the reconstructed points are optimized through a sparse bundle adjustment algorithm.

This chapter is organized as follows: In Section 3.1, the algorithm for reconstructing a scene from multiple views is explained. The elimination of outliers from the reconstruction is discussed in Section 3.2, and the optimization of reconstruction variables is explained in Section 3.3.

3.1 Multiple View Reconstruction

3.1.1 Overview of the Multiple View Reconstruction Algorithm

The procedure of reconstructing a scene from an image sequence can be summarized as follows: Firstly, an initial sparse 3D reconstruction is obtained by using the first two views of the image sequence, as explained in the previous chapter. This initial reconstruction provides the camera matrices of the first two views, features detected and matched in the first two views, and the locations of the 3D points corresponding to these matches.

Having obtained an initial two-view reconstruction, the third view is added to the reconstruction as follows: First, features are detected on the third image, and the features belonging to the second and the third images are matched. Then, this set of matches is searched in order to find out the correspondences that have already been reconstructed. Such features in the third image, which are associated with a 3D point, allow the deduction of the camera projection matrix of the third view, which is explained in more detail in Section 3.1.2. Once the projection matrix of the third view is computed, then the matches between the second and the third views which have not been associated with a 3D point are triangulated to reconstruct the corresponding points in 3-space. Adding these points to the reconstruction, the same procedure is repeated for the other views as well to insert them into the reconstruction. A block diagram explaining this process is given in Figure 3.1.

Throughout the procedure depicted in Figure 3.1, the reconstruction is always achieved in a projective manner, and when the algorithm terminates, this whole projective reconstruction is converted to a metric one by stratifying all 3D points and camera matrices via the rectifying homographies obtained from the initial two-view reconstruction. In other words, given N images, the projective camera

matrices $\mathbf{P}^1_{\mathbf{p}}$ and $\mathbf{P}^2_{\mathbf{p}}$ belonging to the first and second views are chosen as $\mathbf{P}^1_{\mathbf{p}} = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}^2_{\mathbf{p}} = [[\mathbf{e}']_\times \mathbf{F} \mid \mathbf{e}']$, where $\mathbf{F}$ is the fundamental matrix between these views and $\mathbf{e}'$ is the epipole of the second view. Then the projection matrices $\mathbf{P}^3_{\mathbf{p}}$, … , $\mathbf{P}^N_{\mathbf{p}}$ and the scene points $\{\mathbf{X}^i_{\mathbf{p}}\}$ are determined projectively, as summarized in Figure 3.1. Once the projective reconstruction is completed, the metric projection matrices $\mathbf{P}^1_{\mathbf{m}}$, … , $\mathbf{P}^N_{\mathbf{m}}$ and the stratified scene points $\{\mathbf{X}^i_{\mathbf{m}}\}$ are obtained as

$$\begin{aligned} \mathbf{P}^j{}_{\mathbf{m}} &= \mathbf{P}^j{}_{\mathbf{p}}\mathbf{H} \\ \mathbf{X}^i{}_{\mathbf{m}} &= \mathbf{H}^{-1}\mathbf{X}^i{}_{\mathbf{p}} \end{aligned} \tag{3.1}$$

where $\mathbf{H}$ is the rectifying homography computed from the camera calibration matrix $\mathbf{K}$ as discussed in Section 2.4.3.

The multiple view reconstruction algorithm studied in this work does not concentrate on the problem of determining the order in which the images in the input image sequence are inserted into the reconstruction. It is assumed that the image sequence is already ordered in a natural and reasonable way to allow the detection of a sufficient number of matches between consecutive images. Actually, a satisfactory reconstruction requires not only that the consecutive images share a considerable amount of common viewpoint to ease feature matching, but also that their viewing cameras should not be too close to each other, which may lead to an ill-conditioned reconstruction problem. A possible input image sequence for this reconstruction method may be a set of video frames extracted from the video at suitable intervals.

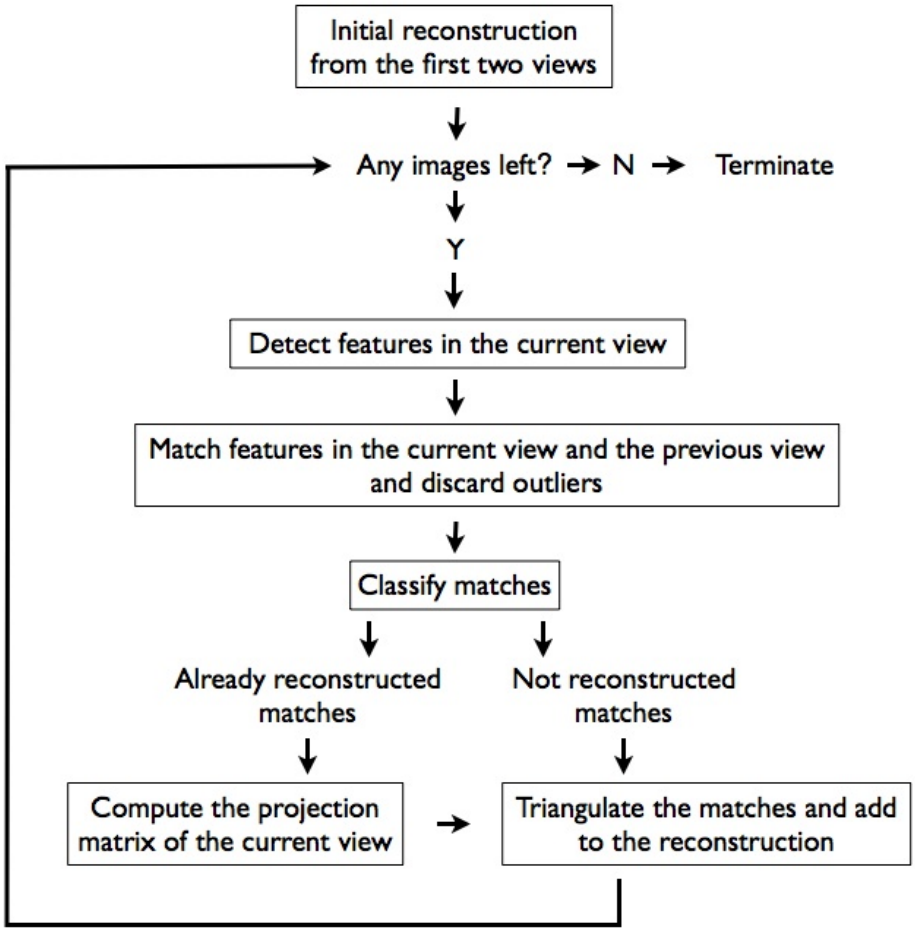


Figure 3.1: Outline of the multiple view reconstruction algorithm

3.1.2 Estimation of the Projection Matrix from 3D Points and Their 2D Projections

As explained in Section 3.2.1, the insertion of a new frame to the reconstruction requires the estimation of the camera projection matrix of a view from a set of already reconstructed matches. In this section, the computation of the projection matrix is explained in detail.

The detection of already reconstructed matches provides a set of 3D points $\{\mathbf{X}_i\}$ and their 2D projections $\{\mathbf{x}_i\}$ on the view which is being integrated into the reconstruction. Denoting the camera projection matrix of this view by $\mathbf{P}$, each pair of 3D-2D points $(\mathbf{X}_i, \mathbf{x}_i)$ must satisfy $\mathbf{P}\mathbf{X}_i = \mathbf{x}_i$. This equation can be rearranged to obtain [1]

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & -w_i\mathbf{X}_i^{\mathrm{T}} & v_i\mathbf{X}_i^{\mathrm{T}} \\ w_i\mathbf{X}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -u_i\mathbf{X}_i^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{bmatrix} = \mathbf{0}, \tag{3.2}$$

where $\mathbf{P}^{i\mathrm{T}}$ is the i^{th} row vector of $\mathbf{P}$, and the point $\mathbf{x}_i$ in homogeneous coordinates is

$$\mathbf{x}_i = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix}.$$

As (3.2) indicates, each projection pair $(\mathbf{X}_i, \mathbf{x}_i)$ defines two linear equations in the entries of $\mathbf{P}$. Since $\mathbf{P}$ has 12 entries, it can be calculated from at least 6 of such pairs. It is also possible to compute $\mathbf{P}$ from all data with a least-squares solution. However, as in the fundamental matrix estimation problem, this is not a preferred approach, since the erroneous data such as mismatched pairs would contaminate

the solution. Therefore, the **P** matrix is estimated from minimal samples, i.e. samples containing 6 projection pairs, by using RANSAC.

In the estimation of the projection matrix from 3D-2D projection pairs with RANSAC, the inlier and outlier pairs are determined with respect to the *reprojection error*. Given a pair ($\mathbf{X}_i$, $\mathbf{x}_i$) and a projection matrix **P**, the reprojection error E_i is defined as

$$E_i = d(\mathbf{x}_i, \mathbf{P}\mathbf{X}_i)^2, \tag{3.3}$$

where d denotes the distance in 2D Euclidean space. If the reprojection error is below some predefined threshold, the pair is regarded as consistent with the projection matrix and classified as an inlier, and otherwise, as an outlier.

When discussing fundamental matrix estimation in Section 2.3.2, it was explained that the normalization of coordinates is required for improving the conditioning of the equation system. A similar normalization is performed in the computation of the projection matrix as well. The centroid of the 2D and 3D points in the sample are moved to the origin by a translation, and then the points are scaled to make the RMS distance to the centroid $\sqrt{2}$ for 2D points, and $\sqrt{3}$ for 3D points. The transformation that results in these normalizations is given in (2.28) for the 2D image points. Similarly, the 3D scene points can be normalized through the transformation

$$\mathbf{X}_{\mathrm{n}} = \mathbf{T}_3\mathbf{X}, \qquad \mathbf{T}_3 = \begin{bmatrix} \frac{\sqrt{3}}{d_{RMS}} & 0 & 0 & -m_X\frac{\sqrt{3}}{d_{RMS}} \\ 0 & \frac{\sqrt{3}}{d_{RMS}} & 0 & -m_Y\frac{\sqrt{3}}{d_{RMS}} \\ 0 & 0 & \frac{\sqrt{3}}{d_{RMS}} & -m_Z\frac{\sqrt{3}}{d_{RMS}} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

where $\mathbf{X}$ is the original point, $\mathbf{X}_{\mathrm{n}}$ is the normalized point, d_{RMS} is the RMS distance to the centroid, and m_X, m_Y and m_Z are the average of respectively the X, Y and Z coordinates of the 3D points in the sample. Denoting the transformation matrix for the 2D points as $\mathbf{T}_2$, the projection matrix $\mathbf{P}$ of the original problem is computed from the normalized projection matrix $\mathbf{P}_{\mathrm{n}}$ by the back-transformation

$$\mathbf{P} = \mathbf{T}_2^{-1}\mathbf{P}_{\mathrm{n}}\mathbf{T}_3. \tag{3.5}$$

The procedure for estimating the projection matrix from a set of projection pairs is summarized as follows:

Algorithm Overview: Computation of the Projection Matrix with RANSAC

Input: A set of 3D-2D projection pairs (**X**,**x**) for an image
Output: The camera projection matrix **P** for the image

1. Initializations:
  a. Set the number of iterations made to 0 (k=0).
  b. Set the upper bound for the number of iterations to an initial value (k_{MAX}=M).

2. While (k<k_{MAX})
  a. Choose a random sample set.

b. Normalize the 2D points in the sample as in (2.28) and the 3D samples as in (3.4).

c. Construct an equation system by using (3.2) and compute the projection matrix.

d. Denormalize the projection matrix by the back-transformation in (3.5).

e. Calculate the reprojection error for all projection pairs and determine the inliers and outliers.

f. If the current model gives the highest number of inliers so far, update the model and the upper bound for the number of iterations k_{MAX}.

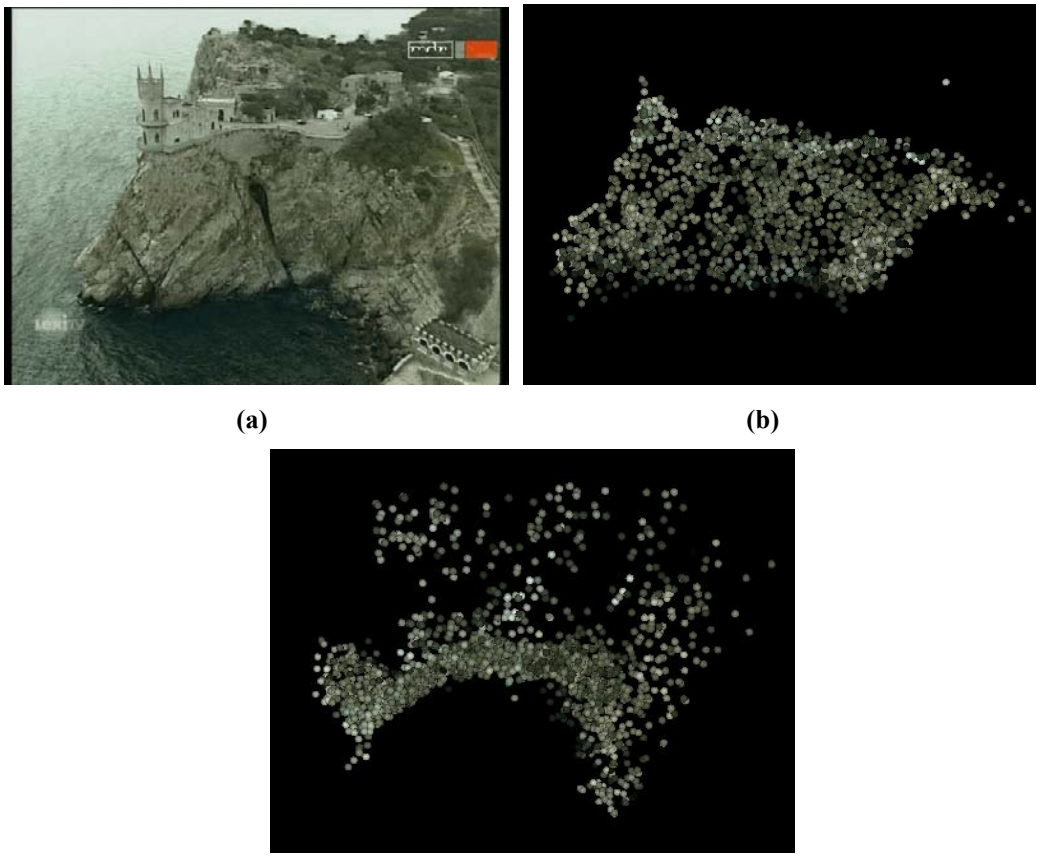
3. Refine the projection matrix iteratively from all inliers.

3.1.3 Experimental Results

The studied multiple view reconstruction method is tested on several image sequences, all of which are extracted from video data.

In Figure 3.2, the reconstruction results for the *Krimm* sequence are presented. The original video is recorded from TV broadcast and the reconstruction algorithm is run on 8 frames that are selected from the video at equal intervals of 10 frames. A sample video frame is given in Figure 3.2(a), and the front and top views of the reconstructed point cloud are shown respectively in (b) and (c).

Another reconstruction experiment is achieved for the *Cevo* sequence, which is captured with a hand-held camera in an indoor environment. Reconstruction is performed on 10 frames extracted from the video at equal intervals of 50 frames. A sample frame is given in Figure 3.3(a), whereas (b), (c) and (d) show respectively the front, side and top views of the reconstructed scene.



(a) **(b)**

(c)

Figure 3.2: Reconstruction results for the *Krimm* sequence: **(a)** A sample frame from the video **(b)** Front view of the reconstructed scene **(c)** Top view of the reconstructed scene

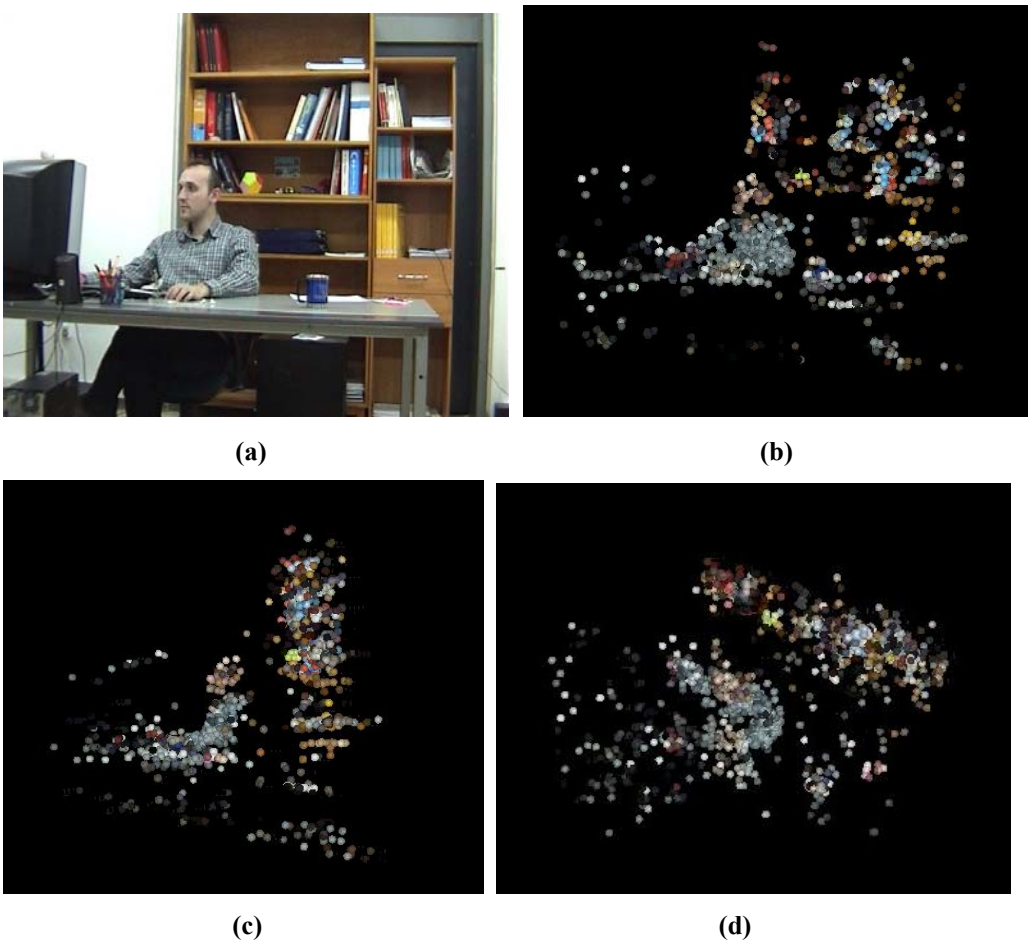
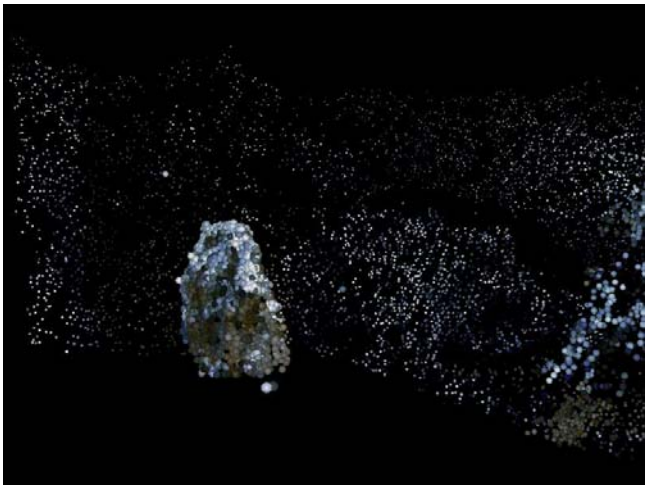
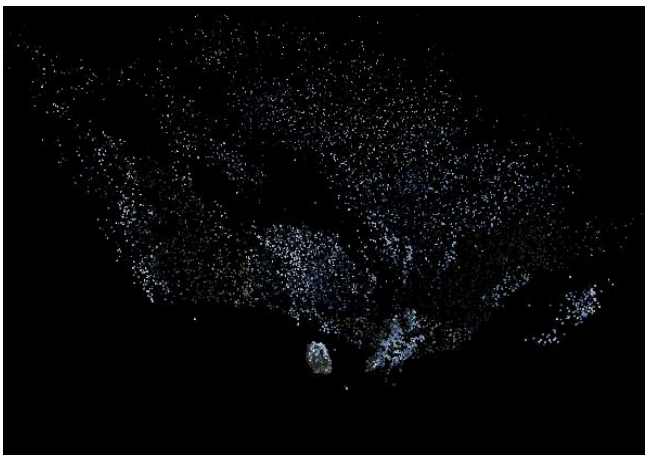


Figure 3.3: Reconstruction results for the *Cevo* sequence: **(a)** A sample frame from the video **(b)** Front view of the reconstructed scene **(c)** Side view of the reconstructed scene **(d)** Top view of the reconstructed scene

(a)



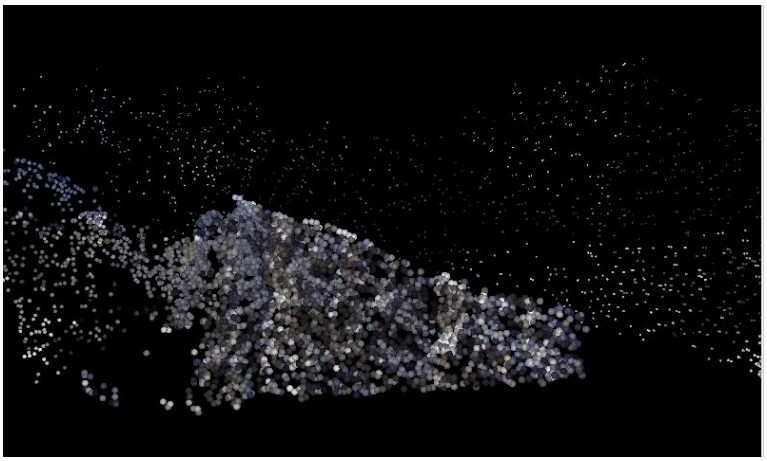
(b)



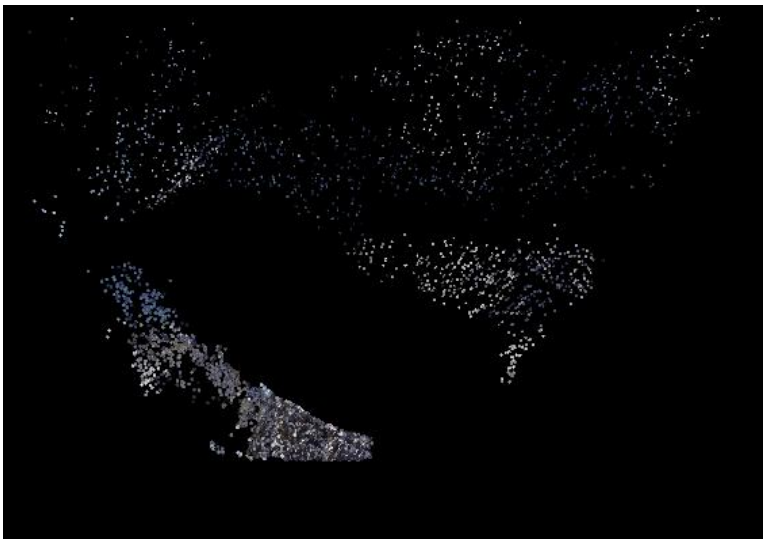
(c)

Figure 3.4: Reconstruction results for *Planet Earth I*: **(a)** Sample video frame **(b)** Front view of the reconstructed scene **(c)** Above view of the reconstructed scene

(a)



(b)



(c)

Figure 3.5: Reconstruction results for *Planet Earth II*: **(a)** Sample video frame **(b)** Close front view of the reconstructed scene **(c)** Overall upper view of the reconstructed scene

Finally, two different scenes from the documentary video *Planet Earth* are reconstructed with the studied algorithm. The first part, *Planet Earth I*, is reconstructed from 19 video frames taken with equal intervals of 30 frames, and the results are presented in Figure 3.4. A sample frame is given in (a), a close front view of the reconstructed scene is shown in (b), and an overall upper view of the scene is given in (c).

The second part taken from the same documentary, *Planet Earth II*, is reconstructed using 11 frames from the video, again at equal intervals of 30 frames. A sample video frame showing the scene is given in Figure 3.5(a). A rather detailed front view of the reconstructed scene is shown in (b), and an overall view of the scene is given in (c). The calibration information for this content is obtained by assuming the principle point to be at the center of the image plane, and adjusting the focal length f by trial and error.

3.2 Removal of Outliers from the Reconstruction

The studied method for sparse reconstruction is capable of extracting 3D point clouds from image sequences effectively. However, the reconstructed point cloud might sometimes be contaminated with erroneous points which actually do not belong to the scene, which usually happens when feature detection and matching is performed with rather loose thresholds.

The occurrence of erroneous points may be due to the inaccuracies in feature detection, false matching, and several other error sources during the estimation of fundamental matrices and projection matrices. A common source of error is mismatches satisfying the epipolar constraint by coincidence, which in turn lead to the reconstruction of erroneous points that yield small reprojection and Sampson errors. Therefore, such points are hard to detect and eliminate via conventional methods.

In this work, it has been aimed to achieve the detection and elimination of such undesired erroneous points within the sparse reconstruction in two different ways. One of these methods is based on defining a suitable cost function in terms of the reconstructed point locations and image intensity functions, and then estimating the correct classification of the reconstructed points as inliers and outliers by minimizing the cost function, which is explained in Section 3.2.1. On the other hand, the other method tries to achieve outlier removal by detecting the spiky points that are away from their neighbours, and then searching the image intensity correlations around these points to determine whether the point is an inlier or not. This second method is discussed in Section 3.2.2.

3.2.1 Outlier Elimination via Optimization

The proposed method for the removal of outliers via optimization is based on the following observations: Given a reconstructed 3D point cloud representing a scene and two images of the scene;

i) The intensity values at the reprojections of a valid 3D point on two image planes must be close to each other with the *Lambertian* surface assumption.

ii) If the reprojection of a valid 3D point in one of the images and a neighbouring point in the same image have similar intensity values, then these two points must belong to the same part of the scene (same object, or surface, etc.). Therefore, the 3D distance between the two points must be relatively small.

In a manner inspired by the usage of Markov random fields in Bayesian methods for 2D motion estimation [21], these observations can be represented via a cost function. The discrimination of a valid 3D point (inlier) from an erroneous 3D point (outlier) is achieved through the minimization of the following function *f*:

$$f\left(\{o_i\}_{i=1}^{N}\right) = \lambda_1 f_1\left(\{o_i\}_{i=1}^{N}\right) + \lambda_2 f_2\left(\{o_i\}_{i=1}^{N}\right) + f_3\left(\{o_i\}_{i=1}^{N}\right) \tag{3.6}$$

$$f_1\left(\{o_i\}_{i=1}^{N}\right) = \sum_{i=1}^{N}\left[I_2(z_i) - I_1(z_i)\right]^2 \delta(o_i)$$

$$f_2\left(\{o_i\}_{i=1}^{N}\right) = \sum_{i=1}^{N}\delta(o_i)\sum_{z_j \in n(z_i)}\delta(o_j)\left\|z_i - z_j\right\|^2 \delta_{\mathrm{T}}\left(I_1(z_i) - I_1(z_j)\right)\delta_{\mathrm{T}}\left(I_2(z_i) - I_2(z_j)\right)$$

$$f_3\left(\{o_i\}_{i=1}^{N}\right) = \sum_{i=1}^{N} o_i$$

where N is the number of reconstructed points; z_i is the 3D location of the i^{th} point; $I_1(z_i)$ and $I_2(z_i)$ are the intensity values at the reprojection of z_i on the first and second image planes respectively; $n(z_i)$ denotes the set of points whose reprojections are the closest neighbours of the reprojection of z_i; o_i is a Boolean function of z_i such that

$$o_i = \begin{cases} 0 & \text{if } z_i \text{ is an inlier} \\ 1 & \text{if } z_i \text{ is an outlier} \end{cases},$$

and $\delta_{\mathrm{T}}(x)$ is a modified delta function such that

$$\delta_{\mathrm{T}}(x) = \begin{cases} 1 & \text{if } x \text{ is in a predefined neighbourhood of 0} \\ 0 & \text{otherwise.} \end{cases}$$

The classification of points as inliers and outliers is achieved through the minimization of f by searching the optimum configuration of o_i values. The term f_1 in (3.6) constrains the inliers to have reprojections in two images with similar intensities, which corresponds to the condition (*i*). Similarly, the term f_2 is in association with the condition (*ii*), and it requires the 3D distances between two inlier points with neighbouring reprojections to be small, provided that the intensity values at the reprojections of these points are close to each other. The

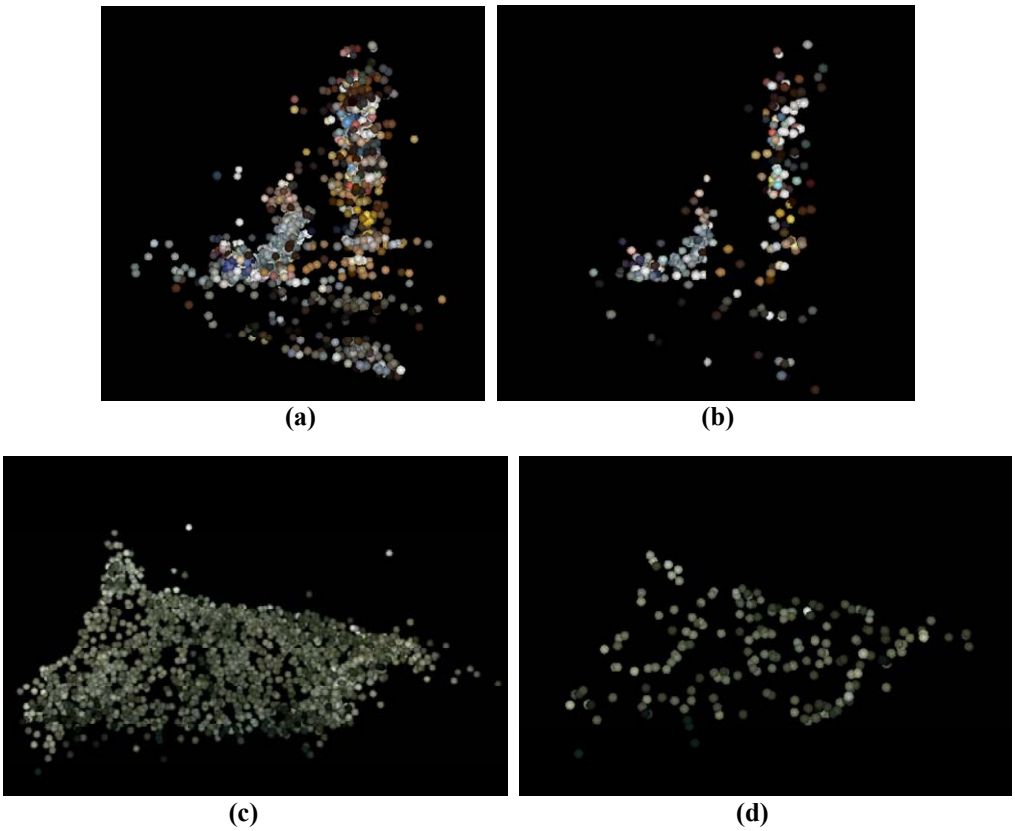


Figure 3.6: Effect of outlier removal via optimization: **(a)** Original noisy reconstruction for *Cevo* sequence **(b)** Reconstruction for *Cevo* after outlier removal **(c)** Original noisy reconstruction for *Krimm* sequence **(d)** Reconstruction for *Krimm* after outlier removal

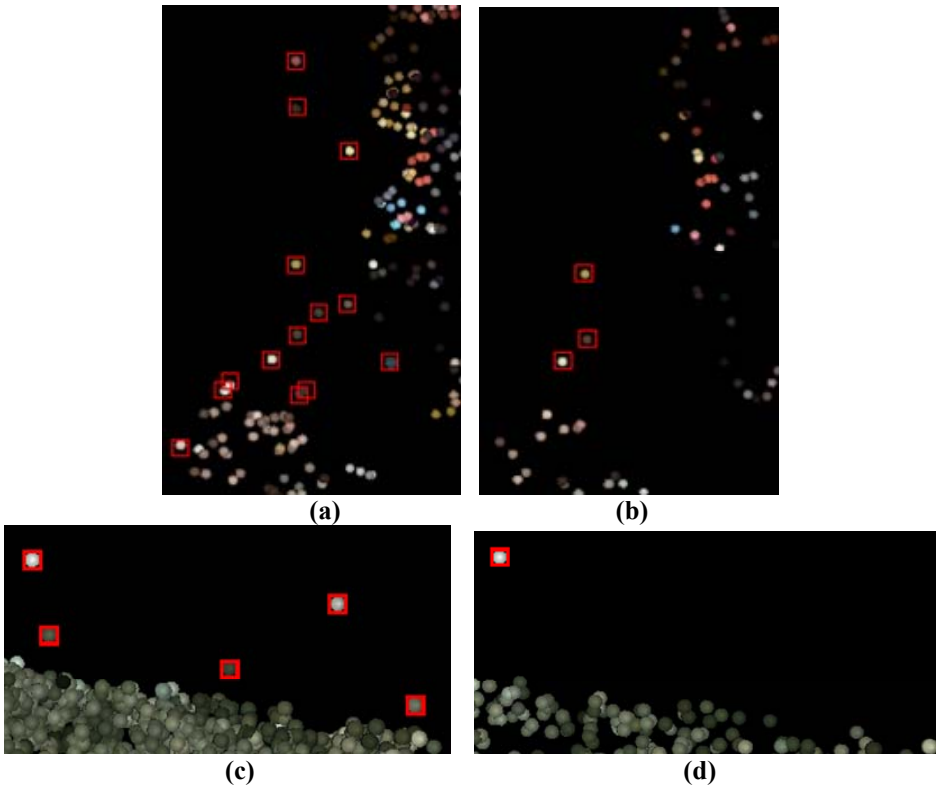


Figure 3.7: Effect of outlier removal via optimization: **(a**) Detailed view from the original point cloud for *Cevo* **(b)** *Cevo* after outlier elimination **(c)** Detail from the original point cloud for *Krimm* **(d)** *Krimm* after outlier elimination

overall cost function f is a weighted sum of f_1, f_2 and f_3, where the term f_3 prevents the algorithm from declaring an unlimited number of points as outliers. Solution to (3.6) is a generalized optimization problem, where all the values of $\{o_i\}$ should be obtained together while minimizing the cost function in (3.6). However, since such an approach is computationally complex, the method of iterated conditional modes (ICM) [21] is utilized during the optimization procedure. ICM minimizes (3.6) in a sequential manner, while minimizing the cost function for a particular o_i value at its each step.

The proposed method for refining a sparse 3D point cloud is tested on the *Krimm* and *Cevo* sequences. The simulation results are given in Figure 3.6. In (a) and (c), the original noisy point clouds obtained from the sparse reconstruction of the scene are given for *Cevo* and *Krimm* respectively, whereas (b) and (d) show the same scenes after the outliers are detected and discarded. In Figure 3.7, typical regions of these scenes are shown in detail. The erroneous points in these regions, shown in red boxes, are marked manually before and after outlier elimination. The results show that the proposed algorithm is capable of eliminating the erroneous and noisy points.

3.2.2 Intensity Correlation Based Outlier Elimination

The second method developed for the removal of outliers from the scene is based on the following observation: In a reconstructed point cloud, the erroneous points usually appear as spikes that cause discontinuities in the distribution of point locations. This encourages the development of an algorithm that detects such spiky points by searching the 3D distances between the reconstructions of neighbouring image points.

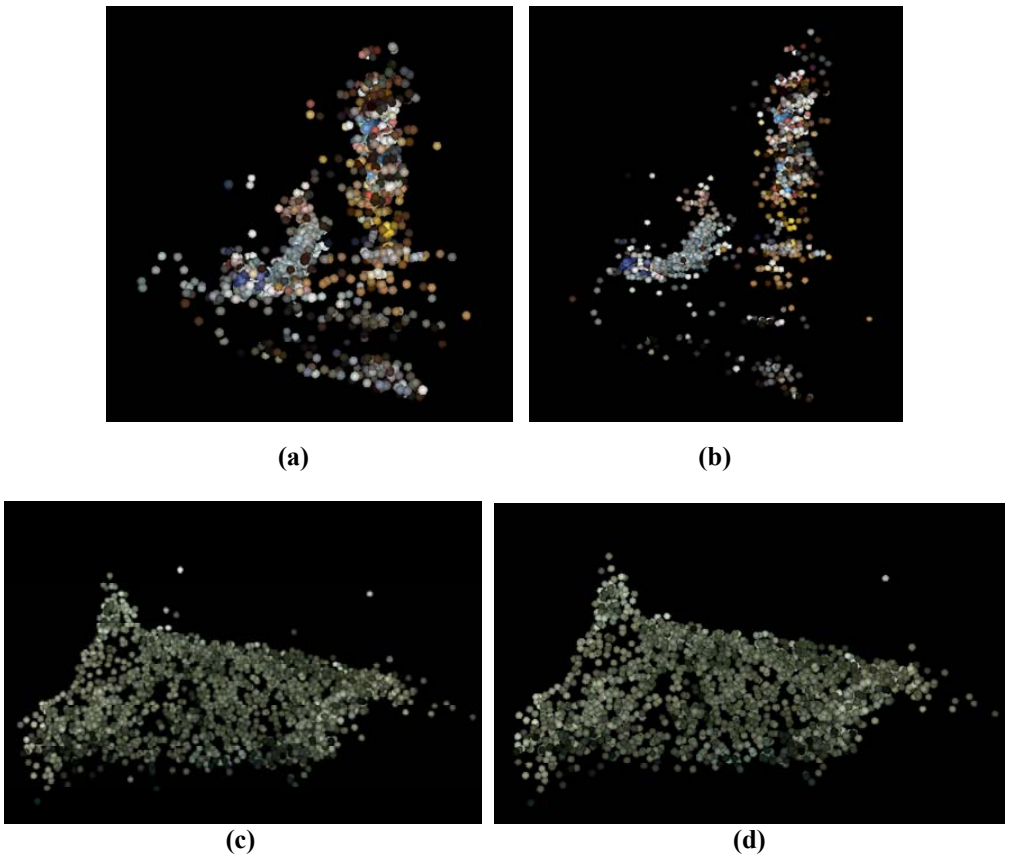


Figure 3.8: Effect of intensity correlation based outlier removal: **(a)** Original noisy reconstruction for *Cevo* sequence **(b)** Reconstruction for *Cevo* after outlier removal **(c)** Original noisy reconstruction for *Krimm* sequence **(d)** Reconstruction for *Krimm* after outlier removal

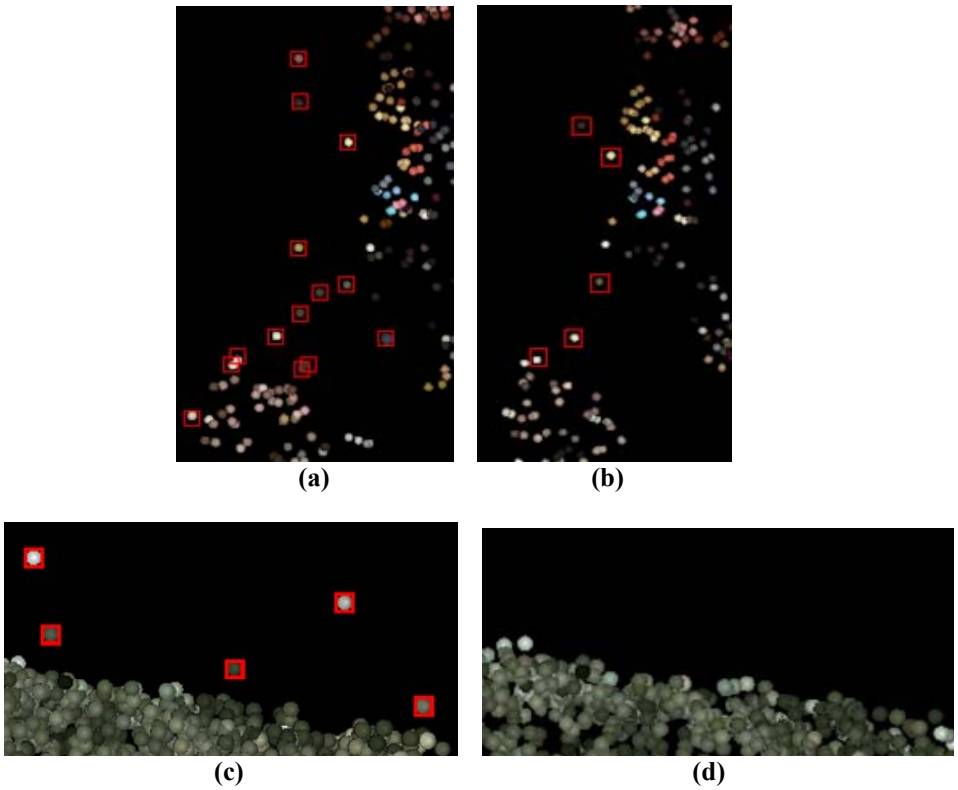


Figure 3.9: Effect of intensity correlation based outlier removal: **(a**) Detailed view from the original point cloud for *Cevo* **(b)** *Cevo* after outlier elimination **(c)** Detail from the original point cloud for *Krimm* **(d)** *Krimm* after outlier elimination

With this approach, erroneous points are detected with the following procedure: Given an image sequence and a reconstructed point cloud, the validity of each scene point is searched by first reprojecting the points on one of the images, and then determining the closest neighbours of the point of interest on the image plane. If the average 3D distance between the point and its neighbours exceeds a predefined threshold, then that point is determined to be a point of discontinuity. On the other hand, a spiky point is not necessarily an erroneous point; therefore, the suspicious points detected as explained are examined further. For this reason, such undecided points are searched by computing the intensity correlations on image patches around the reprojections of these points throughout the whole image sequence. If the image intensity correlation between any two consecutive image patches is below some threshold, then the point is determined to be erroneous and discarded.

The results obtained with this outlier elimination approach are presented in Figures 3.8 and 3.9. In Figure 3.8, the original noisy point clouds for the *Cevo* and *Krimm* sequences are given together with the point clouds after outlier removal. In Figure 3.9, again, some typical scene regions for these reconstructions are shown in detail, where outliers are marked manually before and after outlier removal. The results show that the majority of the outliers are removed efficiently, while inliers are retained.

3.3 Sparse Bundle Adjustment of the Reconstruction Parameters

When the sparse structure of a scene from an image sequence is reconstructed and erroneous points are removed, the last stage is the refinement of the reconstruction via optimization.

The sparse bundle adjustment algorithm, which is presented in [22] and provides a practical implementation of Levenberg-Marquardt algorithm [23] for a large number of reconstruction parameters, is utilized during the optimization of the reconstruction parameters. The optimization variables are the reconstructed point locations $\{\mathbf{X}_i\}$ and the projection matrix parameters $\{\mathbf{P}_j\}$ for each camera; and a cost function that constitutes an approximation of the reprojection error is minimized.

For the implementation of the bundle adjustment algorithm, the camera projection matrices are decomposed into the rotation and translation parameters, and optimization is performed by the adjustment of these parameters. The reasons for representing the camera matrices in terms of the rotation and translation parameters in the optimization are the following:

Firstly, another approach in the parametrization of the projection matrices, such as perturbing the matrix entries independently, is not preferred, since such a practice could lead to degradation in the quality as a metric reconstruction. In other words, the projection matrices obtained at the metric reconstruction step must still conform to the pinhole camera model in the bundle adjustment stage.

Secondly, although the camera projection matrices yielded by the conversion to metric reconstruction must ideally fit the pinhole camera model, in practice, some computational errors, such as those in feature detection or fundamental matrix estimation prevent the final projection matrices from being ideal pinhole camera matrices. Therefore, decomposing the projection matrices to the closest rotation and translation parameters and taking them as the initial estimates in bundle adjustment serves the purpose of idealizing the projection matrices to fit the pinhole camera model and refining these parameters through optimization.

As explained in Section 2.4.1, an ideal pinhole camera matrix is in the form $\mathbf{K}[\mathbf{R}\,|\,\mathbf{t}]$, where **K** is the camera calibration matrix, **R** is a rotation matrix and **t** is a vector representing the translation. However, the projection matrices obtained by metric reconstruction are in the form

$$\mathbf{P}_j = \mathbf{K}\left[\tilde{\mathbf{R}}_j \,|\, \tilde{\mathbf{t}}_j\right],$$

where $\tilde{\mathbf{R}}_j$ is a matrix resembling a rotation matrix but not an ideal one. In order to set the initial estimate parameters of the sparse bundle adjustment algorithm, the matrices $\{\tilde{\mathbf{R}}_j\}$ and $\{\tilde{\mathbf{t}}_j\}$ are obtained by multiplying the projection matrices with $\mathbf{K}^{-1}$ and scaling properly afterwards. Then $\{\tilde{\mathbf{R}}_j\}$ are converted to the nearest rotation matrices $\{\mathbf{R}_j\}$ as discussed in Appendix A. After this conversion, the quaternion representations of the rotation matrices are computed as explained in Appendix B, so that a 4×1 quaternion vector $\mathbf{q}_j$ and a 3×1 translation vector $\mathbf{t}_j$ is extracted from each projection matrix $\mathbf{P}_j$.

The optimization variables in sparse bundle adjustment are the entries of the quaternion and translation vectors for each camera, and the 3D locations of the reconstructed scene points. These parameters are initialized as explained, and then, optimized by the Levenberg-Marquardt algorithm, which is an adaptive optimization algorithm that swings between the steepest descent and Newton algorithms.

Although the camera calibration information is assumed to be known in the reconstruction problem studied in this thesis, it is possible to have some errors in the camera calibration matrix. For example, calibration errors might occur when the available camera calibration matrix is provided by a self-calibration algorithm, or there may be impairments in the physical structure of the camera. Therefore,

the optimization of the camera calibration matrices has also been involved in this refinement stage by adjusting the two entries of the calibration matrix corresponding to the focal length, which are perturbed independently, since these two entries might be different from each other in practical cameras.

Table 3.1: The effects of sparse bundle adjustment and the optimization of internal camera parameters on the reprojection error

	Original Error	Error after the idealization of P matrices	Error after SBA	Error after SBA and optimization of K
Cevo	0.044	2.850	0.263	0.243
Planet Earth I	0.031	4.108	1.452	1.257
Krimm	0.065	46.683	1.154	0.956

The results given in Table 3.1 show the effect of sparse bundle adjustment on the reconstruction. The bundle adjustment algorithm is tested on the reconstructions of the *Krimm*, *Cevo* and the *Planet Earth I* sequences by computing the average reprojection error at several stages. The first column indicates the average reprojection errors calculated with the original reconstructions, which do not conform to the ideal pinhole camera model. The reprojection errors yielded by the idealized projection matrices that fit the pinhole camera model, which are also the initial values of the optimization, are given in the second column. The reprojection errors after optimization are presented in the third and fourth columns, where the values in the third column are obtained by adjusting only the rotation and translation parameters and point coordinates; whereas adjusting the calibration matrix (focal length) as well leads to the errors listed in the fourth column.

The results presented in Table 3.1 suggest the following: The reprojection error is quite small for the original reconstruction. However, there is a significant increase in the reprojection error, when the camera matrices are approximated by the closest pinhole cameras. Starting with these initial errors, the sparse bundle adjustment algorithm, i.e. the optimization of the scene point locations and the rotation and translation parameters of all cameras, decreases the reprojection error to a great extent. Adjusting the focal length parameters together with the bundle adjustment leads to a slight further decrease in the error.

Although the utilized optimization algorithm seems to have a considerable contribution to the refinement of the reconstruction, it must be noted that the Levenberg-Marquardt algorithm is a gradient-descent type algorithm, and in order to have satisfactory results in the optimization stage, the algorithm must start from good initial estimates of the optimization variables, since the cost function, i.e. the total reprojection error, is a complicated function of the adjusted parameters.

CHAPTER 4

MESH-BASED SCENE REPRESENTATION WITH APPLICATIONS TO VIDEO ENCODING

The output of the sparse reconstruction algorithm presented in Chapters 2 and 3 is a 3D point cloud. Although sparse reconstruction provides an efficient tool for estimating the camera poses in an uncalibrated image sequence, its output is not amenable to direct usage in practical applications. In order to convert the sparse reconstruction to a visually more pleasant one, dense reconstruction must be achieved, which may be in the form of a dense depth map, as well as a mesh representation of the scene.

This chapter is devoted to the examination and evaluation of the piecewise planar scene representation algorithm proposed by İmre [28], which aims to obtain a rate-distortion efficient mesh representation of a scene from a sparse point cloud. In Section 4.1, some introductory information about the algorithm and the mesh-based scene representation problem is given, whereas a more detailed examination of the method is presented in Section 4.2.1. Finally, the results of some experiments investigating the performance of the algorithm are discussed in Section 4.2.2.

4.1 Introduction

The mesh representation of a scene is a natural means of upgrading a sparse point cloud to a dense reconstruction. The work of İmre [28] is based on obtaining a piecewise planar approximation of the scene surface, called a mesh, given

irregular samples from the surface, which are actually the 3D points obtained from sparse reconstruction and constitute the vertices of the mesh. The final goal of the algorithm is to construct a textured piecewise planar model of the scene consisting of triangular patches from a given point cloud, where the texture information is also provided by a given image of the scene.

A key consideration in the design of the algorithm is the purpose of achieving this in a rate-distortion efficient way, where the term *rate* is associated with the number of vertices in the mesh structure, and the *distortion* measure is taken as the image intensity error between a known image of the scene and a one predicted from the reconstructed mesh. In such a scene representation problem, there is a natural trade-off between the number of vertices included in the structure, hence the quality of the representation, and data storage limitations. The goal of the algorithm presented in [28] is to find a compromise between these in an efficient and computationally feasible way.

As well as the purpose of achieving dense 3D scene reconstruction, another motivation reason behind the development of such an algorithm is its direct applicability to video coding. Since the method is based on minimizing the error between a known and a rendered image of the scene, it implicitly provides a tool for mesh-based frame prediction in video coding.

4.2 Rate-Distortion Efficient Scene Representation

4.2.1 Algorithm

In this section, the algorithm of the rate-distortion efficient scene representation method presented in [28] is reviewed. Given a 3D point cloud, a reference and a target view of the scene, together with the camera projection matrices belonging to these views; the purpose of the algorithm is to construct a mesh representing

the scene, by adding the input 3D points in a sequential manner to the mesh. A critical aspect of the method is that, at each cycle, it aims to determine the most suitable point among the whole set to add to the mesh, such that the error in the prediction of the target view is minimized.

Before the discussion of the point selection strategy of the algorithm, the procedures for mesh generation and rendering the predicted view must be overviewed. Given a 3D point cloud, the reconstruction of the mesh containing these points requires the determination of the connections between the points. In the studied algorithm, the piecewise planar model of the scene consists of triangular patches, and the construction of the triangles in 3-space is achieved indirectly in the 2D plane as follows: Firstly, the 3D scene points are projected onto the 2D image plane of the reference camera. Then the image region containing the 2D projections is divided into triangles by using the Delaunay triangulation algorithm [29],[30], which forms triangles from given vertices in a manner that the triangles resemble equilateral ones as much as possible. Once the triangulation in the 2D plane is achieved, the construction of the 3D piecewise planar model is obtained easily by lifting the triangulation to 3D, where 2D vertices are substituted with the corresponding 3D scene points. An example of Delaunay triangulation in 2D plane is depicted in Figure 4.1.

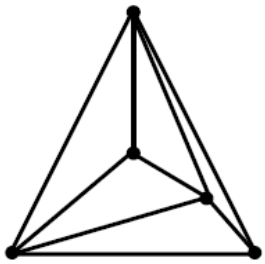


Figure 4.1: Construction of triangles from a set of vertices in a 2D plane by Delaunay triangulation [28]

In order to predict the target view from the mesh and the reference view, the view of the model belonging to the target camera must be rendered. In a general rendering problem, where the reference and target camera matrices, the scene structure and the reference texture map are known, the value of the texture map corresponding to the target view can be obtained at each pixel by joining the camera center of the target view and the corresponding pixel with a line, intersecting the line with the 3D model to find the 3D point projecting to the pixel of interest, and then determining the intensity value of the reference texture map at the projection of this 3D point.

If the scene is represented with a piecewise planar model, then the rendering problem can be approached as a mapping between the reference and the target views defined by a set of *homographies*, since two views of a planar scene are related via a homography [1]. It is possible to compute the 3×3 homography matrix $\mathbf{H}_i$ for each triangular patch, from the camera matrices $\mathbf{P_r}$ and $\mathbf{P_t}$ of the reference and the target cameras, and the plane normal of the patch $\mathbf{n}_i$ [28]. The procedure of rendering the predicted view is illustrated in Figure 4.2.

The rate-distortion efficient scene representation algorithm proposed in [28] operates with a coarse-to-fine approach. The algorithm first constructs an initial mesh with a minimal number of bounding points, and then refines this model in a sequential manner by adding a new vertex (a new 3D scene point) to the model at each iteration. The new vertex is selected with a specific rate-distortion efficient strategy, which will be explained shortly, and then integrated into the mesh structure, by inserting the projection of the vertex to the current 2D Delaunay triangulation corresponding to the 3D mesh model, and then updating the mesh accordingly.

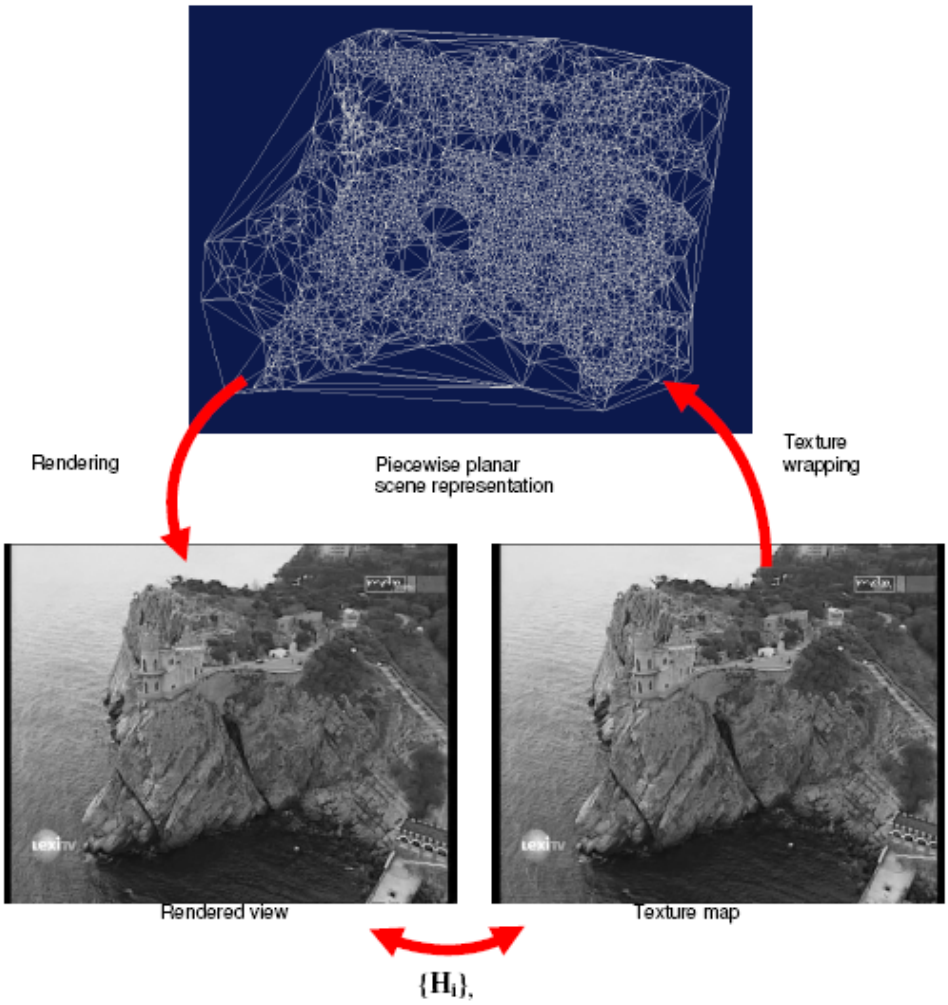


Figure 4.2: The texture map of the target view can be obtained from the texture map of the reference view through a set of homographies, if the scene is represented with a piecewise planar model [28].

Selection of the new vertex is achieved with the following strategy: Since the algorithm aims to operate in a rate-distortion efficient way, the vertex that provides the greatest decrease in the mean square error between the predicted and the known views of the target frame must be selected. Ideally, this requires constructing all possible meshes by trying all candidate vertices, computing the error for all meshes, and then choosing the vertex that yields the minimum error. However, this is computationally infeasible. Instead, the algorithm follows such a procedure: Firstly, the triangular patch where the vertex will be added is determined by computing the total prediction error in the image region associated with each patch, and then selecting the patch with the highest error. After the determination of the patch to be modified, the vertices whose projections lie on the projection of the selected patch are examined. Among these vertices, the one that conforms to the homography defined by the patch least is chosen as the new vertex to be added to the scene representation, where the conformance to the homography is determined by the symmetric transfer error *d* [1],

$$d = (\mathbf{x}_r - \mathbf{H}^{-1}\mathbf{x}_t)^2 + (\mathbf{x}_t - \mathbf{H}\mathbf{x}_r)^2, \tag{4.1}$$

where $\mathbf{H}$ is the homography matrix corresponding to the patch, and $\mathbf{x}_r$ and $\mathbf{x}_t$ are the projections of the 3D point on the reference and the target views respectively.

At each cycle of the algorithm, a vertex is selected from the given input set as explained and the mesh is updated. The update in the scene representation is accepted if it decreases the prediction error. This process is repeated until the bit budget spared for the scene representation is consumed, or the prediction error converges. A final refinement of the reconstruction is possible through nonlinear optimization.

The rate-distortion efficient scene representation algorithm proposed in [28] can be summarized as follows:

Algorithm Overview: Rate-Distortion Efficient Piecewise Planar Scene Representation

Input: A set of 3D scene points, the camera projection matrices and texture maps of the reference and target views of the scene
Output: A piecewise planar representation of the scene

1. Construct an initial mesh with a minimum number of boundary points:

a. Determine the initial boundary points.

b. Project the points to the reference 2D image plane.

c. Perform Delaunay triangulation and lift the triangulation to 3D to obtain the mesh.

d. Determine the prediction error corresponding to each triangular patch.

2. Until the prediction error converges or the bit budget is consumed:

a. Choose the patch with the highest prediction error (the mean square error between the rendered and the known target views)

b. Among the 3D points projecting on the same image region with the selected patch, choose the one which least fits the homography defined by the patch, by using (4.1).

c. Add the selected vertex to the mesh by first inserting its projection to the 2D triangulation, and then lifting the 2D triangulation to 3D.

d. Accept the update in the mesh if the prediction error is decreased.

e. Update the prediction errors belonging to the patches.

3. Refine the scene representation with nonlinear optimization.

4.2.2 Experimental Results

In order to assess the performance of the piecewise planar scene representation algorithm presented in [28], several experiments have been conducted. In these experiments, the vertex selection strategy of the studied algorithm has been compared to some other possible strategies in the sense of rate-distortion performance. In all experiments, the *rate* measure is taken to be the number of vertices used in the mesh structure, and the *distortion* is accepted as the mean square error between the predicted and the known texture maps of the target view.

In the first set of experiments, the performance of the vertex selection strategy of the studied algorithm is compared to that of the random vertex selection strategy. To this aim, for a given input set of 3D points, the mesh is first constructed with the studied algorithm sequentially using the vertex selection approach examined in Section 4.2.1. Then, the mesh is constructed again from the same input set sequentially, but by adding a randomly selected vertex to the mesh at each cycle this time; such that the final mesh is the same as the one yielded by the studied algorithm, whereas intermediate meshes differ.

The random mesh construction experiment is repeated 10 times. The results obtained with the vertex selection approach of the studied algorithm are shown together with the results of random vertex selection approach on the same graphs, where the MSE between the predicted and the known target views is plotted against the number of vertices. The experiment is conducted on the sparse reconstructions of four different data, namely, *Venus*, *Cevo*, *Castle* and *Krimm*. Sample frames from the data are given in Figure 4.3, and the graphs are presented in Figure 4.4. In each graph, the blue curve is yielded by the studied rate-distortion efficient algorithm, whereas the red marks represent the states reached during the intermediate steps of mesh reconstruction with random vertex selection.

The graphs presented in Figure 4.4 suggest that the performance of the studied algorithm is quite acceptable. Actually, if the same experiment was conducted with a *rate-distortion optimal* algorithm, i.e. an algorithm that would always choose the best mesh yielding the smallest prediction error among all possible meshes for a given number of vertices, then all error measurements belonging to the randomly generated meshes would certainly lie above the curve generated by the rate-distortion optimal algorithm. In other words, the curve belonging to a rate-distortion optimal algorithm can be determined uniquely by taking the minimum possible value of the error for each vertex number. If the experiment results given in Figure 4.4 are interpreted with this point of view, it can be concluded that the performance of the studied rate-distortion efficient algorithm is indeed close to that of a rate-distortion optimal one, as the curve belonging to the examined algorithm constitutes the lower boundary for the random trials most of the time.

As explained in Section 4.2.1, in each cycle, the algorithm selects the triangular patch with the worst prediction error, and then among the vertices corresponding to this patch, the vertex with least conformance to the homography defined by the patch is added to the mesh. It is the approach of selecting the triangular patch with the worst error, which is questioned in the second set of experiments. For this reason, the rate-distortion performance of the examined algorithm is compared to that of a modified algorithm, which chooses a random triangular patch at each cycle, and then selects a vertex from this patch with the same strategy, i.e. chooses the vertex with least conformance to the homography. Again, the experiment is repeated 10 times with the modified algorithm for each data, and the resulting graphs are presented in Figure 4.5.

The plots presented in Figure 4.5 suggest the following: Firstly, as in the previous experiment, the studied rate-distortion efficient algorithm has been seen to outperform the modified algorithm that selects the triangular patches randomly, as the curve corresponding to the studied algorithm resides below the marks

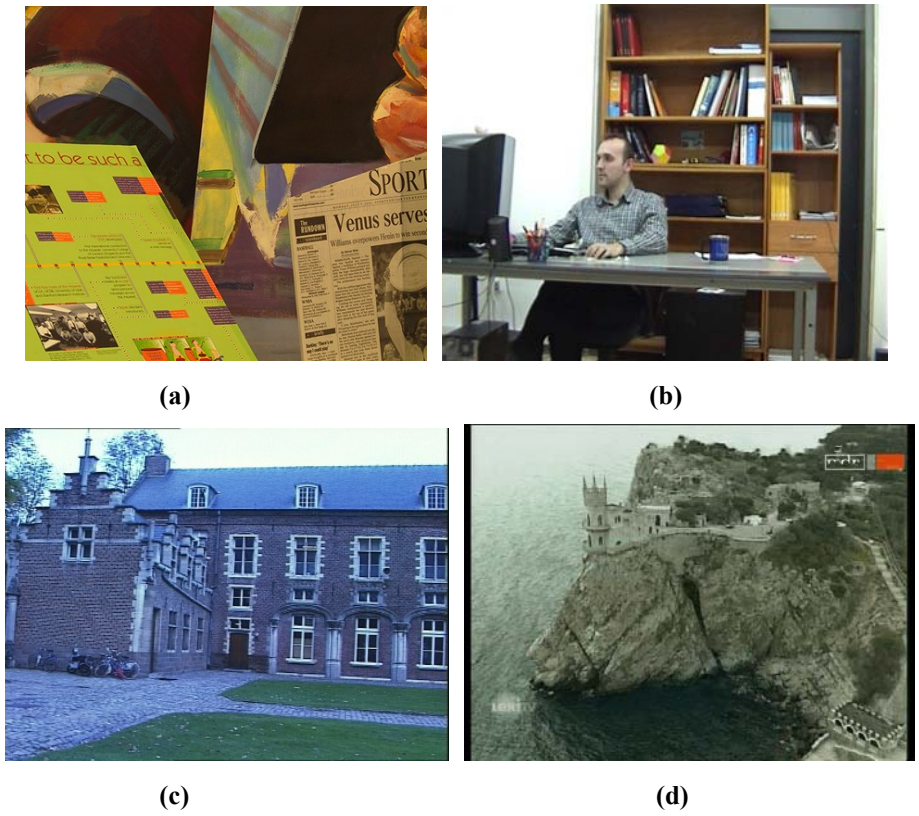
generated by the modified algorithm most of the time. This confirms that the approach of selecting the triangular patch with the largest total prediction error in the examined algorithm is a proper strategy. Another conclusion can also be drawn from this experiment, if the plots obtained in Figures 4.4 and 4.5 are compared. A fall in the general distribution of the red marks is easily noticeable. Assuming that the random vertex selection in the first experiment (presented in Figure 4.4) is equivalent to selecting a random triangle, and then selecting a random vertex from this triangle, it can be deduced that the selection of the vertex with least conformance to the homography, which makes the difference between the two experiments, is a suitable approach.

The strategy of choosing the vertex that worst fits the homography is actually investigated further in another set of experiments. In this part, the performance of the examined rate-distortion efficient algorithm is compared to those of three other modified algorithms, where the modified algorithms select the triangular patch with the largest prediction error as in the examined algorithm, whereas the vertex selection from the chosen patch is achieved in each one of the algorithms by selecting the vertex with respectively the second, third, and fourth largest symmetric transfer errors. The resulting rate-distortion plots are presented in Figure 4.6.

A natural expectation about this experiment is to have the rate-distortion performances of the compared strategies ordered such that the one that chooses the vertex with the highest error performs the best, and the one choosing the vertex with the fourth highest error performs the poorest. The results for *Venus* and *Cevo* are in general consistent with this expectation, whereas the plots obtained for *Castle* and *Krimm* are not. However, an explanation for this situation might be the following: *Venus* is a synthetic data where the camera projection matrices are perfectly available. *Cevo* is a real data, however, it is captured in a controlled environment, and the camera calibration matrix is available, therefore, the reconstruction achieved with this data can be considered to be reliable as well.

On the other hand, *Krimm* is a data captured from TV broadcast, and the camera calibration information for *Krimm* is obtained via a self-calibration algorithm, which might have caused the reconstruction to be affected by some calibration errors. Finally, although the calibration parameters are available for *Castle*, the scene contains some dominant planes, and it is known that such prominent planes in the scene structure are likely to cause errors in fundamental matrix estimation, consequently in the reconstruction [14]. Therefore, a possible explanation for the discrepancy between the graphs presented in Figure 4.6 might be that in (c) and (d), during the mesh generation and vertex selection phases, the algorithm might have been misled by some error sources, which may have affected the reconstruction of the point cloud given as input to the algorithm.

In a final experiment, the performance of the rate-distortion efficient algorithm is compared to another one, which tries every possible candidate for the new vertex and chooses the one yielding the smallest prediction error at each cycle This modified algorithm can be considered to be a rate-distortion optimal one, in the sense that it selects the best possible vertex at each iteration. The experiment is conducted only on *Venus*, and the results are presented in Figure 4.7. The plots reveal that the algorithm choosing the optimal vertex performs better than the rate-distortion efficient algorithm in terms of the prediction error, which is an expected result. However, the rate-distortion curve yielded by the examined algorithm is fairly close to that of the optimal one, while its computational cost is significantly lower. Therefore, the overall performance of the studied rate-distortion efficient scene representation algorithm can be considered to be quite acceptable.



(a) **(b)**

(c) **(d)**

Figure 4.3: Sample frames from the sequences used in the experiments: **(a)** *Venus* **(b)** *Cevo* **(c)** *Castle* **(d)** *Krimm*

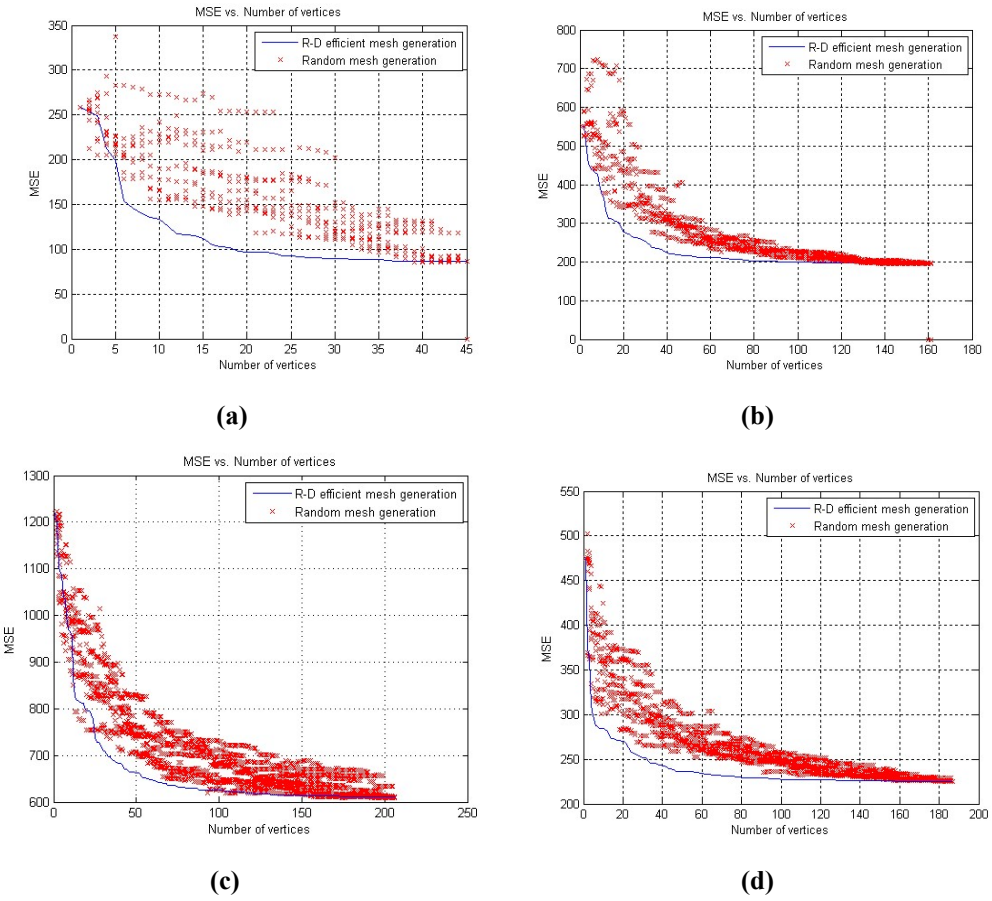


Figure 4.4: Prediction errors obtained during the sequential construction of mesh-based scene representations for **(a)** *Venus* **(b)** *Cevo* **(c)** *Castle* **(d)** *Krimm*, with random vertex selection

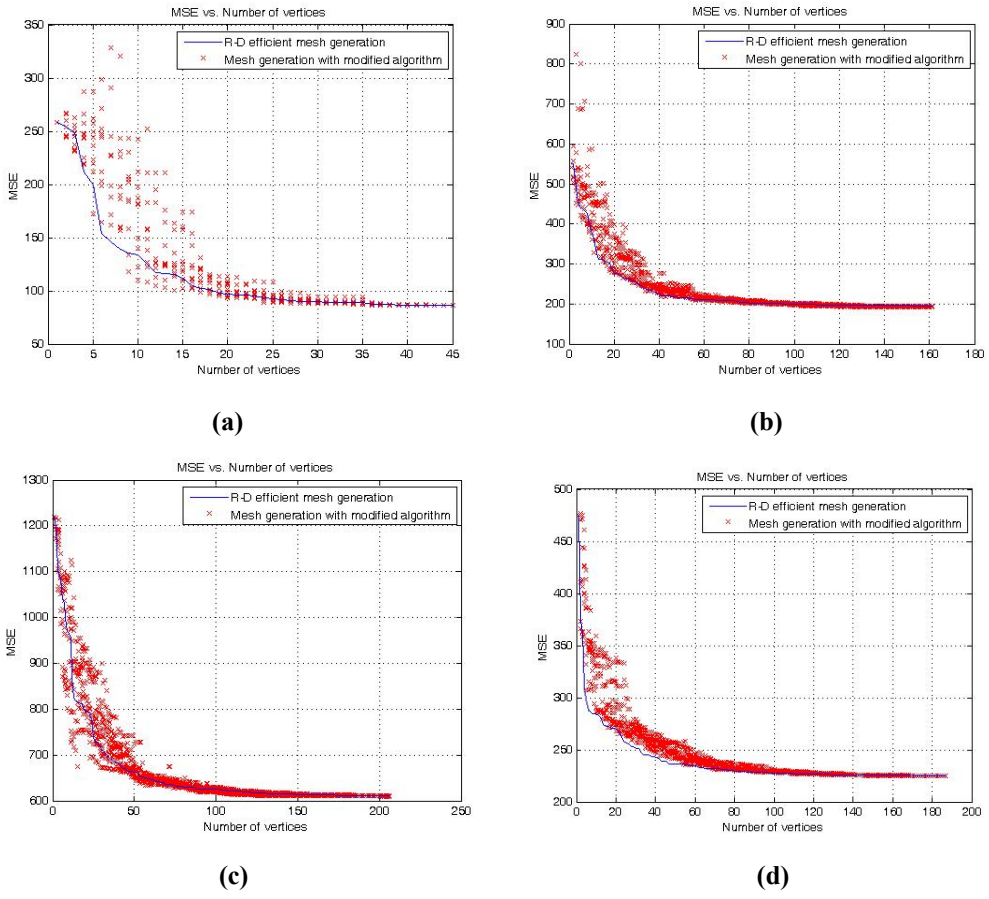


Figure 4.5: Prediction errors obtained during the sequential construction of mesh-based scene representations for **(a)** *Venus* **(b)** *Cevo* **(c)** *Castle* **(d)** *Krimm*, with random triangular patch selection

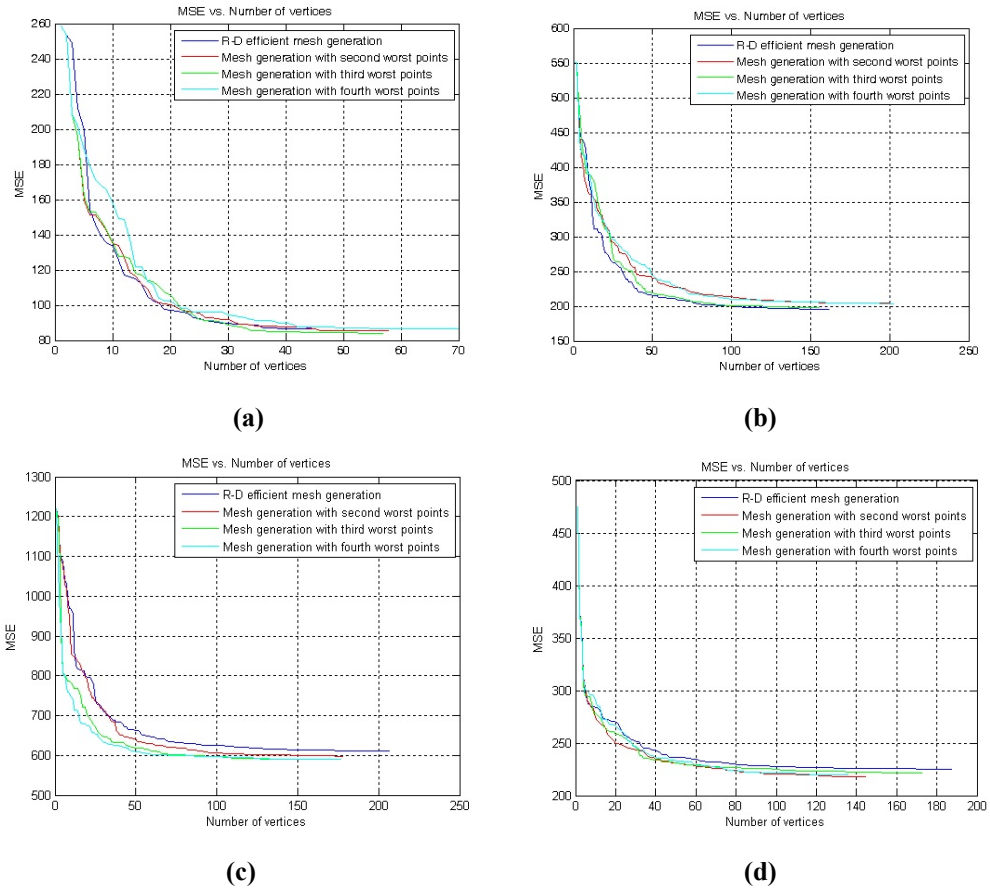


Figure 4.6: Prediction errors obtained during the sequential construction of mesh-based scene representations for **(a)** *Venus* **(b)** *Cevo* **(c)** *Castle* **(d)** *Krimm*, with different mechanisms of vertex selection from patches.

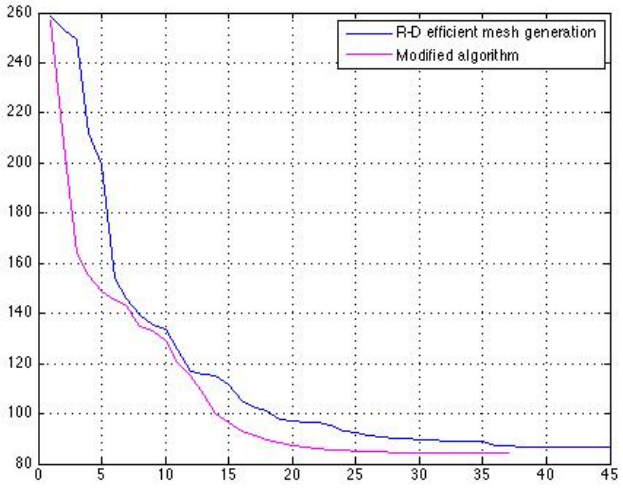


Figure 4.7: Prediction errors obtained during the sequential construction of mesh-based scene representation for *Venus* with optimal and rate-distortion efficient vertex selection algorithms.

CHAPTER 5

CONCLUSIONS

Retrieving 3D information from 2D image content is a challenging research subject in computer vision. This thesis addresses the problem of the extraction of a 3D point cloud from a calibrated image sequence.

Being the initial step of sparse 3D reconstruction, feature detection and matching is achieved with the method of Scale Invariant Feature Transform (SIFT) [4] in this work. In order to evaluate the performance of SIFT, some experiments have been conducted for the comparison of SIFT, Harris corner detector [3] and KLT tracker [20]. These algorithms have been compared with respect to the amenability of the matches they provide to the epipolar geometry estimation. Among these three algorithms, KLT tracker has resulted to produce the largest number of matches consistent with the estimated model (inliers), whereas the ratio of the correct matches among all matches is the highest for SIFT. The smallest average Sampson error per inlier is also yielded by SIFT. The outcomes of the experiments confirm that SIFT is not only capable of accurate feature point localization, but also robust feature matching.

Another basic stage in reconstruction is the estimation of the epipolar geometry relating the two views, which is represented with the fundamental matrix. 7-point

[1] and 8-point [11] algorithms are two major methods for the computation of the fundamental matrix from correspondences. These two algorithms have been tested through experiments, where the searched criteria are the inlier/total match number ratios, Sampson error per inlier and the number of iterations made in RANSAC. The two algorithms have turned out to give close results from these aspects. However, the performance of the 7-point algorithm tends to be slightly better than that of the 8-point algorithm in general. Considering this result together with the fact that 7-point algorithm computes the F-matrix from exactly 7 correspondences, whereas the 8-point algorithm allows more correspondences as input, a reasonable strategy could be to utilize the 7-point algorithm during the iterations of RANSAC, and refine the final model by recalculating the F-matrix from all inliers with the 8-point algorithm.

An important issue in estimating the epipolar geometry from correspondences is the exploitation of a suitable sampling strategy to allow the computation of the F-matrix both effectively and accurately. RANSAC [12] and PROSAC [13] are two algorithms developed for this purpose. Again, the experiments have been conducted with several data in order to compare the performances of the two algorithms. As a consequence of these comparative tests, it has been concluded that the two algorithms perform almost the same as far as the accuracy and validity of the computed model is concerned. However, the major difference between the algorithms is their computation load. The experimental results suggest that at high values of the inlier/total match number ratio, RANSAC terminates faster, however, as the ratio of outliers among all matches increases, PROSAC brings significant computational savings over RANSAC. As a conclusion, in practical problems, it seems to be more advantageous to use PROSAC, if the input set of feature matches is likely to be contaminated with errors. However, RANSAC might be more preferable if the matches are accurate enough.

In the proposed solution to the multiple-view sparse reconstruction problem, the reconstruction of the scene and the cameras is achieved up to a projective transformation at first, and then transformed to a metric reconstruction through a rectifying homography computed from the camera calibration matrix. Although ideally the metric reconstruction of the cameras must conform to the pinhole camera model, the experiments with real data has shown that the computed metric reconstruction does not agree with the pinhole camera model thoroughly, but constitutes a close approximation, which is due to computation errors, inaccurate measurements, etc. Even if the original output of the reconstruction algorithm yields fairly small reprojection errors, it has been seen that the substitution of the camera matrices with their idealized approximations might lead to a considerable increase in the reprojection error.

This problem is remedied to an extent by the utilization of a sparse bundle adjustment algorithm, which takes the idealized but inaccurate reconstruction as the initial estimate, and optimizes the reconstruction parameters. The experimental results show that the exploitation of the sparse bundle adjustment algorithm highly improves the reprojection error, and that it is even possible to reduce the error slightly more by including the optimization of some internal camera parameters in the algorithm as well. However, since the sparse bundle adjustment method is based on a gradient-descent type of optimization algorithm, the initial estimate is critical for the progress of optimization, as the algorithm might miss the optimum solution by getting trapped into a local minimum depending on the starting point.

Another challenge in reconstruction is the occurrence of erroneous points in the reconstructed scene, caused by several error sources such as mismatches, and computation and measurement errors, the accumulations of which are observable especially for long image sequences. Two different methods have been proposed for the removal of outliers from the scene, one of which is based on the minimization of a cost function defined in terms of the reconstructed points and

image intensity functions, whereas the other algorithm aims to eliminate the outliers by detecting spiky points in the reconstruction and searching intensity correlations around these points. Experiments have shown that both methods are capable of discarding outliers effectively. However, the intensity correlation based algorithm might be more preferable if the reconstructed scene is in general correct but contains a few points with prominent errors, whereas the optimization based algorithm might perform better in scenes with distributed errors and noise.

Finally, the piecewise planar scene representation algorithm proposed by İmre [28] is examined, and experiments have been conducted to evaluate its performance. The experimental results have shown that the algorithm is quite promising in the sense of rate-distortion efficiency; and its performance has been seen to be close to that of a rate-distortion optimal one, while the computational complexity of the algorithm is fairly acceptable.

APPENDIX A

The problem of finding the nearest rotation matrix to a given matrix can be formulated as follows [26]: For $\mathbf{A} \in \mathbb{R}^{3\times3}$, the nearest matrix with orthogonal columns to **A** is the matrix **A**+**E**, where

$$\mathbf{E} = \arg\min_{\mathbf{E}} \left\{ \|\mathbf{E}\| \mid \mathbf{E} \in \mathbb{R}^{3\times3}, (\mathbf{A}+\mathbf{E})^{\mathrm{T}}(\mathbf{A}+\mathbf{E}) = \mathbf{I} \right\}. \tag{A.1}$$

In [27], Fan and Hoffman has shown that for $\mathbf{A} \in \mathbb{R}^{n\times n}$, where **A** has the polar decomposition **A**=**UH**, any matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ with orthogonal columns satisfies $\|\mathbf{A} - \mathbf{U}\| \le \|\mathbf{A}-\mathbf{Q}\|$. Therefore, the minimization problem in (A.1) can be solved by obtaining the polar decomposition of **A**.

The polar decomposition is closely related to the singular value decomposition in the following way:

$$\mathbf{A} = \mathbf{WSV}^{\mathrm{T}} = (\mathbf{WV}^{\mathbf{T}})(\mathbf{VSV}^{\mathbf{T}}) = \mathbf{UH}, \tag{A.2}$$

where $\mathbf{WSV}^{\mathrm{T}}$ is the singular value decomposition, and **UH** is the polar decomposition of **A**. Hence, the nearest rotation matrix to an 3×3 matrix **A** is the matrix $\mathbf{U}=\mathbf{WV}^{\mathrm{T}}$.

APPENDIX B

The set of quaternions is a four-dimensional vector space over real numbers, with the operations addition, multiplication by a real number and a special quaternion multiplication. Quaternions provide a compact representation for the rotation of vectors in 3-space, where the unit quaternion **q** corresponding to a rotation of θ degrees around the axis $\hat{\mathbf{e}} = (e_x, e_y, e_z)$ is

$$\mathbf{q} = \begin{bmatrix} e_x \sin(\theta/2) \\ e_y \sin(\theta/2) \\ e_z \sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix}. \tag{B.1}$$

Given a rotation matrix **R**, the unit quaternion $\mathbf{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^{\mathrm{T}}$ representing the rotation can be obtained as:

$$\begin{aligned} q_4 &= \pm\frac{1}{2}\sqrt{1 + \mathbf{R}_{11} + \mathbf{R}_{22} + \mathbf{R}_{33}} \\ q_1 &= \frac{1}{4q_4}\left(\mathbf{R}_{32} - \mathbf{R}_{23}\right) \\ q_2 &= \frac{1}{4q_4}\left(\mathbf{R}_{13} - \mathbf{R}_{31}\right) \\ q_3 &= \frac{1}{4q_4}\left(\mathbf{R}_{21} - \mathbf{R}_{12}\right) \end{aligned} \tag{B.2}$$

Similarly, given a unit quaternion $\mathbf{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^{\mathrm{T}}$, the rotation matrix **R** corresponding to **q** is given by

$$\mathbf{R} = \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2(q_1q_2-q_3q_4) & 2(q_1q_3+q_2q_4) \\ 2(q_1q_2+q_3q_4) & 1-2q_1^2-2q_3^2 & 2(q_2q_3-q_1q_4) \\ 2(q_1q_3-q_2q_4) & 2(q_1q_4+q_2q_3) & 1-2q_1^2-2q_2^2 \end{bmatrix}. \quad \text{(B.3)}$$

REFERENCES

[1] R. Hartley, A. Zisserman, *Multiple View Geometry*, Cambridge University Press, UK, 2003.

[2] M. Pollefeys, "Visual 3D Modeling From Images", Nov. 2002. [Online]. Available: http://cs.unc.edu/~marc/tutorial.pdf. [Accessed: Aug 5, 2008].

[3] C. G. Harris and M. Stephens, "A combined corner and edge detector", *Proceedings of 4th Alvey Vision Conference*, pp. 147-151, 1988.

[4] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, vol. 60, No. 2, pp. 91-110, 2004.

[5] A. P. Witkin, "Scale-space filtering", *International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, pp. 1019-1022, 1983.

[6] T. Lindeberg, "Feature Detection with Automatic Scale Selection", *International Journal of Computer Vision*, vol. 30, No. 2, 1998.

[7] T. Lindeberg, "Scale-space theory: A basic tool for analysing structures at different scales", *Journal of Applied Statistics*, vol. 21, No. 2, pp. 224-270, 1994.

[8] K. Mikolajczyk, "Detection of local features invariant to affine transformations", Ph.D. thesis, Institut National Polytechnique de Grenoble, France, 2002.

[9] E. Tola, “Multiview 3D Reconstruction of a Scene Containing Independently Moving Objects”, M.S. thesis, Middle East Technical University, Ankara, Aug. 2005.

[10] H. C. Longuet-Higgins, “A computer algorithm for reconstructing a scene from two projections”, *Nature*, vol. 293, pp. 133-135, Sept. 1981.

[11] R. I. Hartley, “In Defence of the 8-Point Algorithm”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.19, No. 6, pp. 580-593, 1997.

[12] M. Fischler and R. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography”, *Communications of the ACM*, vol. 24, No. 6, pp. 381-385, 1981.

[13] O. Chum, J. Matas, “Matching with PROSAC: Progressive Sample Consensus”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 220-226, 2005.

[14] O. Chum, T. Werner, J. Matas, “Two-View Geometry Estimation Unaffected by a Dominant Plane”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol.1, pp. 772-779, 2005.

[15] P. H. S. Torr and A. Zisserman, “MLESAC: A new robust estimator with application to estimating image geometry”, *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.

[16] P. D. Sampson, “Fitting conic sections to ‘very scattered’ data: An iterative refinement of the Bookstein algorithm”, *Computer Vision, Graphics, and Image Processing*, vol. 18, pp. 97-108, 1982.

[17] Z. Zhang, R. Deriche, O. Faugeras, Q. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry", *Artificial Intelligence* 78, pp. 87-119, 1995.

[18] C. Schmid and A. Zisserman, "The geometry and matching of curves in multiple views", *Proceedings of European Conference on Computer Vision*, pp. 394-409, Springer-Verlag, June 1998.

[19] S. Maybank, *Theory of Reconstruction from Image Motion*, Springer-Verlag, Berlin Heidelberg, 1993.

[20] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", *DARPA Image Understanding Workshop*, pp 121-130, 1981.

[21] M. Tekalp, *Digital Video Processing*, Prentice Hall, 1995.

[22] M. Lourakis, A. Argyros, "The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm", ICS/FORTH, Technical Report No. 340, Aug. 2004.

[23] M. Lourakis, "A Brief Descriptipon of the Levenberg-Marquardt Algorithm Implemented by levmar", Feb 2005. [Online]. Available: http://www.ics.forth.gr/~lourakis/levmar/levmar.pdf. [Accessed: Aug 19, 2008].

[24] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops and R. Koch, "Visual Modeling with a Hand-Held Camera", *International Journal of Computer Vision*, vol. 59, No. 3, 2004.

[25] E. İmre, S. Knorr, B. Özkalaycı, U. Topay, A. Alatan, T. Sikora, "Towards 3-D Scene Reconstruction from Broadcast Video", *Signal Processing: Image Communication 22*, pp 108-126, 2007.

[26] N. Higham, "Matrix Nearness Problems and Applications", *Applications of Matrix Theory*, pp. 1-27, Oxford University Press, 1989.

[27] K. Fan, A. J. Hoffman, "Some metric inequalities in the space of matrices", *Proc. Amer. Math. Soc.*, vol. 6, pp. 111-116, 1955.

[28] E. İmre, "Prioritized 3D Scene Reconstruction and Rate-Distortion Efficient Representation for Video Sequences", Ph.D. thesis, Middle East Technical University, Ankara, Aug. 2007.

[29] O. Devillers, S. Meiser, M. Teillaud, "Fully Dynamic Delaunay Triangulation in Logarithmic Expected Time per Operation", *Computational Geometry in Theory and Application,* vol. 2, No. 2, pp. 55-80, 1992.

[30] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, *2nd Ed.*, Springer Verlag, Germany, 2000.