Fundamental Problems in Computational Video

By
Anthony David Whitehead

A doctoral thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

April 2004

Preface and Acknowledgements

This thesis represents a subset of my efforts over the last several years. The thesis describes a subset of the work carried out at the School of Computer Science, Carleton University, specifically that work that includes computational video problems as a core focus. I am very grateful to Prosenjit Bose for his encouragement and infectious zeal and constant guidance to improve the theoretical writing style. The work that follows has been greatly enriched by Robert Laganiere, whose implicit understanding of computer vision and comments have resulted in fruitful explorations. Thanks is also owed to Gerhard Roth for providing initial exposure to the field of projective vision for which much of this thesis revolves.

I would like to also extend thanks to all whom, via personal correspondence, have provided useful guidance and insights on a variety of different topics. Specifically, Franz Oppacher, David Lowe, Mark Pollefeys, Patrick Morin, and Richard Hartley have all provided interesting commentary.

A great deal of thanks must go to both family and friends who make the journey towards this end an enjoyable one. Thanks particularly go to Darcy, Jason, Mike, Pat, and Veronique for their influence, and helping me be under it. Furthermore, my sister Justine, thanks for all the interesting talk and always picking up the tab, and most fundamentally thanks to Tanya for keeping me focused.

I am grateful to Carleton University and Nortel for providing my funding.

Abstract

Problems in computer vision and computational video often make certain assumptions about the input data. For example, structure from motion algorithms assume a baseline of minimal configuration and reconstruction problems often assume a known corresponding feature set and calibration parameters. Often it is the case that these assumptions present difficult problems in themselves.

This thesis examines problems that maintain a common thread of video data. Often coined computational video, this emerging field presents a number of interesting problems that often fall into the assumptions of other research areas. Specifically, we address suitable baseline selection for structure from motion as well as an automated system that solves the correspondence problem in large number of cases. We also address a means for automatically computing intrinsic camera parameters for long video sequences and we examine a method for synchronizing multiple video streams. Furthermore, we address the problem of accurate segmentation of commercial video streams for subsequent use in video databases and search facilities.

Each of the problems addressed in this thesis are often assumed to be solved in the presentation of other research problems such as 3D reconstruction, video abstracting and database population. Each of the proposed solutions provides benefits to the research community by providing tools and/or novel algorithms that address these often assumed sub-problems. Furthermore, the findings presented in this thesis remove a number of constraints that are generally placed on these types of problems.

Contents

Preface and Acknowledgements ii
Abstract iii
Contents iv
List of Figures viii
Guide to Notation x
1 Introduction 2
1.1 From Computer Vision to Computational Video 2
1.2 Motivation 3
1.3 Thesis Outline and Contributions 4
2 Background 8
2.1 Computational Video 8
2.2 Camera Models 9
2.4 Overview of Projective Geometry 11
2.4.1 Projective n-spaces 12
2.4.2 Collineations 12
2.4.3 The Projective Plane (P^2) 13
2.4.3.1 Points and Lines 13
2.4.3.2 Lines at Infinity 13
2.4.3.3 Pencils of Lines 14
2.4.3.4 Conics 14
2.4.4 The Projective Space P^3 14
2.4.4.1 Quadrics 14
2.4.5 Strata of Geometries 15
2.4.5.1 Projective Strata 15
2.4.5.2 Affine Strata 16
2.4.5.3 Metric Strata 16
2.4.5.4 Euclidean Strata 17
2.4.5.5 Strata Review 17
2.4.5.6 Changing Strata 18
2.5 Camera Calibration 20
2.5.1 Intrinsic 20
2.5.2 Extrinsic Calibration 21
2.6 Planer Transformations 22
2.7 Stereo Vision 25
2.7.1 Correspondence Problem 25
2.7.2 Epipolar Geometry 25
2.7.3 Essential Matrix 27
2.7.4 Fundamental Matrix 27
2.8 Robust Methods for Computing the Epipolar Geometry 29
2.8.1 RANSAC 30

2.9 Three View Geometry 30
2.9.1 The Trifocal Tensor 31
2.9.2 Constraints on the Tensor 32
2.9.3 Robust Computation of the Tensor 33
2.10 N-View Geometry 33
2.11 Feature Tracking 34
2.11.1 The Feature Tracking Equations 34
2.11.2 Good Features to Track 36
3 Computing Camera Positions from Uncalibrated Video/Image Sequences 38
3.1 Introduction 38
3.2 Processing Steps 43
3.2.1 Selecting Frames that are well suited 43
3.2.1.1 Motion Estimation and Feature Tracking 44
3.2.1.2 Salient Frame Extraction 45
3.2.1.3 Salient Frame Extraction Results 46
3.2.2 Finding corners/interest points 48
3.2.3 Matching Corners 49
3.2.4 Local consistency filtering 50
3.2.5 Computing the fundamental matrix 51
3.2.6 Guided matching 52
3.2.7 Computing putative triple correspondences 52
3.2.8 Computing the trifocal tensor 53
3.2.9 Computing the 3D information 53
3.3 Experiments 54
3.4 Conclusions and Discussions 60
4 Segmenting Video Sequences 62
4.1 Introduction 62
4.2 Additional Background 63
4.2.1 Quantifying Inter-frame Differences 64
4.2.1.1 Individual Pixel Differences 64
4.2.1.2 Intensity/color histograms 65
4.2.1.3 Edge based features 65
4.2.2 Classifying Differences as cuts and non-cuts 66
4.2.2.1 Global threshold 66
4.2.2.2 Adaptive threshold 66
4.3 Feature Tracking for Quantifying Dissimilarity 67
4.3.1 Feature Tracking 68
4.3.2 Pruning False Tracking 70
4.4 Automatically Determining a Linear Discriminator 73
4.4.1 Density Estimation 74
4.4.2 Computational Considerations 77
4.4.3 Non-Overlapping Distributions 78
4.4.4 The Candidate Sets (Overlapping Distributions) 78
4.5 Experiments 80
4.5.1 Comparison Metrics 80
4.5.2 Experimental Results 82

4.5.3 Processing Speed 85
4.5.4 The Effect of Feature Selection 86
4.5.5 Known Problem Areas 87
4.6 Conclusions and Discussion 89
5 Autocalibration from the Fundamental Matrix 90
5.1 Introduction 90
5.2 Autocalibration via Equal Eigenvalues 94
5.2.1 Single Image Pairs 94
5.2.2 Multiple Image Pairs 94
5.3 Autocalibration via Kruppa's Equations 95
5.3.1 Single Image Pairs 95
5.3.2 Multiple Image Pairs 96
5.4 The Evolutionary Approach 97
5.4.1 Dynamic Hill Climbing 98
5.4.2 Mutating coordinate frames 99
5.4.3 Exploiting local optima 99
5.4.4 Coverage of Search Space 100
5.4.5 Autocalibration Algorithm 101
5.5 Degeneracy 102
5.5.1 Handling Degeneracy 103
5.6 Experiments 104
5.7 Conclusions and Discussion 110
6 Synchronizing Multiple Video Sequences 113
6.1 Introduction 113
6.1.1 Additional Background 114
6.2 Problem Formalization 116
6.2.1 The synchronization problem 116
6.3 Recovery of the synchronization 122
6.3.1 Computing Camera Geometry from the Static-CCS 123
6.3.2 Generating the Trajectory Images 124
6.3.3 Gross approximation of synchronization via trajectory images 126
6.3.4 Refinement via maximal geometric consensus 128
6.3.5 Synchronization in the face of erroneous geometries 130
6.3.6 Algorithm Review: Synchronize 130
6.4 Experimental Results 131
6.4.1 Synthetic Data 131
6.4.2 Real Data 132
6.5 Practical Considerations 137
6.6 Conclusions and Discussion 138
7 Conclusions and Open Problems 139
7.1 Summary 139
7.2 Future Work 140
A Exact Match counts for PVT Example Sets 142
B The Portable Image Library (PIL) 148
C The Projective Vision Toolkit (PVT) 149
Bibliography 150

Index 156

List of Figures

FIGURE 2.1: THE PINHOLE CAMERA MODEL 10
FIGURE 2.2: THE PINHOLE CAMERA WITH A VIRTUAL IMAGE PLANE 10
FIGURE 2.3: CROSS RATIO OF 4 LINES IN P^2 16
FIGURE 2.4: SHAPE DISTORTIONS FOR EACH GROUP OF TRANSFORMATIONS IN 3D 18
FIGURE 2.5: INTRINSIC PARAMETERS 21
FIGURE 2.6: EXTRINSIC PARAMETERS FOR A GENERIC CAMERA IN THE WORLD 22
FIGURE 2.7: TWO VIEWS OF THE SAME SCENE 23
FIGURE 2.8: TRANSFORMATION OF LEFT IMAGE TO THE VIEW POINT OF THE RIGHT IMAGE 24
FIGURE 2.9: RIGHT VIEW FROM FIG. 2.4 SUPERIMPOSED ONTO FIG 2.5 24
FIGURE 2.10: EPIPOLAR GEOMETRY OF TWO CAMERAS WITH FIXED X DISPARITY 26
FIGURE 2.11: EPIPOLAR GEOMETRY OF TWO CAMERAS IN AN ARBITRARY POSITION 26
FIGURE 2.12: THE TWO TYPES OF CONJUGATE PAIR ERRORS 29
FIGURE 2.13: TRIFOCAL GEOMETRY 31
FIGURE 3.1: SYSTEM FOR GOING FROM VIDEO TO 3D CAMERA POSITIONS 40
FIGURE 3.2**:** EVERY 2ND EXTRACTED FRAME FROM EXAMPLE SEQUENCE 48
FIGURE 3.3: ALL FRAMES FROM EXAMPLE SEQUENCE 48
FIGURE 3.4 SELECTION 9 OF IMAGES FROM VIDEO SEQUENCE 55
FIGURE 3.5 RECONSTRUCTED CAMERA POSITIONS AND POINTS FROM A VIDEO CAMERA 55
FIGURE 3.6 3 SAMPLES FROM THE CMU-BIGHOUSE SEQUENCE, FRAMES 1,6 & 11. 56
FIGURE 3.7 RECONSTRUCTED CAMERA POSITIONS AND FEATURE POINTS 57
FIGURE 3.8 RECONSTRUCTED CAMERA POSITIONS WITH OVERLAPS HIGHLIGHTED 58
FIGURE 3.9 ORIGINAL IMAGE WITH FEATURES AND CAMERAS OVERLAID 58
FIGURE 3.10 EXAMPLE IMAGE WITH OVERLAID CAMERA POSITIONS AND POINTS 59
FIGURE 4.1: HARD CUT BETWEEN FRAMES 206 AND 207 OF A VIDEO SEQUENCE. 63
FIGURE 4.2: SEQUENCE FROM THE MOVIE PSYCHO 67
FIGURE 4.3: DIAGRAM OF SYSTEM TO COMPUTE CUTS 67
FIGURE 4.4**:** END RESULT FOR FEATURE TRACKING OVER SEVERAL FRAMES. 69
FIGURE 4.5: THREE CONSECUTIVE FRAMES FROM A SEQUENCE. 71
FIGURE 4.6: THE RESULTS OF SQUARING. 72
FIGURE 4.7**:** (A) NON-OVERLAPPING DISTRIBUTIONS, (B) OVERLAPPING DISTRIBUTIONS 74
FIGURE 4.8: DETAILED EXAMINATION OF KERNEL FUNCTIONS. 76
FIGURE 4.9: TRIANGULAR KERNEL AT VARIOUS BANDWIDTHS (WINDOW SIZES) 77
FIGURE 4.10: (A) ORIGINAL PDF (B) MODIFIED FIRST DERIVATIVE (G(X)) 79
FIGURE 4.12: DENSITY ESTIMATION GRAPH. 83
FIGURE 4.14: DIGITIZATION ARTIFACTS 87
FIGURE 4.15: EXAMPLES OF PROBLEM SITUATIONS. 88
FIGURE 5.1: SCATTER PLOT OF 2D SEARCH SPACE GENERATED BY SDRS. 100
FIGURE 5.2: DEGENERATE EPIPOLAR GEOMERTY 103
FIGURE 6.6: REQUIRED GEOMETRIES FOR A 3 CAMERA SYNCHRONIZATION 124
FIGURE 6.7: INITIAL IMAGE, TRAJECTORY IMAGE 125
FIGURE 6.8: TRAJECTORY IMAGE (ENLARGED) FOR A 3 SECOND INTERVAL 126

FIGURE 6.9: 3 TRAJECTORY IMAGES WITH OBVIOUS COINCIDENT POINTS OF INFLECTION. 127
FIGURE 6.10: EPIPOLAR LINE AND TRAJECTORIES. 128
FIGURE 6.11: COMPUTING FRAME VIA EPIPOLAR AND TENSOR TRANSFER. 129
FIGURE 6.12: 3 SYNCHRONIZED (ROWS), CONSECUTIVE FRAMES (COLUMNS) 133
FIGURE 6.13: SELECTED SYNCHRONIZED FRAMES 134
FIGURE 6.14: SELECTED SYNCHRONIZED FRAMES 135

Guide to Notation

Throughout the paper, we will use a common convention for notation. In Table 1 below, a list of common notation is provided

c	**:**	The optical center of the image plane in pinhole camera
C	**:**	The optical center of the focal plane in pinhole camera
D	**:**	The Euclidean transformation matrix, this is a 4x4 matrix
E	**:**	The essential matrix, this is a 3x3 matrix
F	**:**	The fundamental matrix, this is a 3x3 matrix
f	**:**	Intrinsic camera parameter, focal length
H	**:**	The homography matrix, this is a 3x3 matrix
K	**:**	The intrinsic camera parameters, this is a 3x3 matrix
l	**:**	A line in homogeneous form, this is an (N+1)D vector
M	**:**	A point in 3D space, this is a 3D vector
m	**:**	A point in an image, this is a 2D vector
P	**:**	The projection matrix (3D to 2D), this is a 3x4 matrix
R	**:**	The Rotation matrix, this is a 3x3 matrix
s	**:**	Scalar factor
t	**:**	The translation vector, this is a 3D vector
$\mathfrak{J}$	**:**	Trifocal Tensor, this is a 3x3x3 cube operator
$\mathbf{U_0}$	**:**	Intrinsic camera parameter, center of projection X
$\mathbf{V_0}$	**:**	Intrinsic camera parameter, center of projection Y
$\boldsymbol{\alpha}$	**:**	Intrinsic camera parameter, aspect ratio
$\boldsymbol{\theta}$	**:**	Intrinsic camera parameter, sensor skew

Chapter 1

Introduction

1.1 From Computer Vision to Computational Video

Research into computer vision has been ongoing for many decades, with origins dating back to the late 50's and early 60's. Due to limited computing power, a lack of understanding, and experience with the complexity of three dimensions, most early work in computer vision concentrated on what today are largely considered 2D problems such as fingerprint processing [1], optical character recognition (OCR) [2], and satellite imagery [3]. Other recent areas of research include the use of computer vision techniques in practical problems like retrieval from image databases, content-based image and video compression, and face recognition to name a few. However, a lot of this work is largely a matter of applying already well-established computer vision methods to new problems but require some level preprocessing of the input data to allow the use of known methods.

A consequential step of the early research was the application of computer vision techniques for interpreting aerial photographs and satellite imagery. Here the images are typically very large and much more complex in their contents and variety, but still are mostly 2D in nature (aside from some small 3D effects like shadows cast by tall buildings or bridges crossing rivers). Satellite imagery is still handled using mostly planar methodologies due to the distance between features being minute in comparison to the distance of the camera to the earth's surface [4]. Work on these types of images dominated computer vision research throughout the seventies. However, some significant work was done on images of simple 3D scenes of the so-called "Blocksworld", a domain of painted toy blocks on a plain tabletop. Still today, many researchers use a blocksworld domain to verify effectiveness and test algorithms.

In the eighties, and still today, the main challenge of computer vision research is attempting to deal more fully with the 3D world, and with the movement of sensors. A lot

of work was supported and influenced by the Autonomous Land Vehicle (ALV)[1] Project in the United States [5], but its origins appear earlier. The ALV was intended to be a robot vehicle that could drive itself around the countryside, guided by computer vision algorithms and other sensors. Within the past decade, the ALV project has added video cameras as a standard sensor to be used on the vehicle [6].

As the video camera plays such an important role in our day to day lives, it is not likely that we will see the video camera diminish as a data acquisition source for some time to come. The volume of video data that has resulted since the advent of the video camera far exceeds that of still image photography and creates a new set of unique and interesting problems that need resolution. If there is any one thing that characterizes the recent trends and direction of computer vision research today, it is the terms dynamic (active) vision and computational video. Researchers are moving away from the implicit view that a vision system is merely a stationary recipient of passive images, realizing that as part of a robot, a vision system will actively move about and explore while interacting with its (dynamic) environment. This new point of view has inspired research in the area of dynamic vision recently, leading to robotics and computational video research labs with an emphasis on video processing research becoming prevalent.

1.2 Motivation

There are a number of reasons for dynamic vision based video processing problems. Primarily (for the author) is that they are interesting practical problems with theoretical backgrounds that combine elements of computer vision; artificial intelligence, efficiency and optimization.

It has often been said that computer vision research is sometimes an exercise in finding the right data set, however, as techniques mature and become standardized for solving certain problems, the challenge always remains in ensuring that unconstrained inputs can be made suitable for the mature algorithms. Until recently it was thought that little work could be done without having the metrics of a calibrated stereo vision system.

[1] A complete list of publications related to the ALV project since 1985 can be found a the ALV website at Carnegie Mellon University (http://www.ri.cmu.edu/labs/lab_28_pubs.html)

Just over a twenty years ago, the Essential matrix was introduced [7], followed shortly by the fundamental matrix [8][9] and not long after that, robust methods for determining the fundamental matrix [10][11] were introduced offering the capability to take a simple, uncalibrated vision system, and perform some machine vision tasks. This work progressed to the introduction of 3 view geometry [12][18]. Since then, much research has been focused on autocalibration and metric reconstructions with uncalibrated stereo rigs. With all these advances, there has been little concentration on video data and the unique problems associated that come with it.

Specifically, this thesis examines techniques that are used as a precursor to the application of well-known photogrammetry and content based video retrieval algorithms. Due to the volume of data contained in a video sequence it is impractical, if not impossible, to effectively apply a standard technique to each and every frame. Furthermore, many algorithms are not well suited for the close proximity of adjacent images produced by the high frame rates in video cameras. Currently, there is still some reliance on techniques such as manual resolution of the correspondence problem or non-linear computational techniques that do not scale up to volume of images found in a typical video sequence. Therefore, this thesis provides some solutions to these types of problems faced by all computer vision researchers who will use video data as an input. The main contributions of this thesis are outlined next.

1.3 Thesis Outline and Contributions

Hopefully the reader is now convinced that the problem of computational video processing for computer vision tasks is both interesting and important. The remainder of this section outlines the chapters and the research contributions of this thesis.

Chapter 2 – Background

This chapter is used as a vehicle for covering certain background information that is necessary for the remainder of the work presented. Theory of multiple view geometry applied to computer vision is outlined in this chapter. Concepts such as homographies, the Essential and Fundamental matrices, camera calibration and the trilinear tensor are

given. Finally we provide some coverage of feature tracking in video sequences as it forms a fundamental primary part in the solution of many computational video problems.

Chapter 3 – From Video Sequences to 3D Camera Positions

The correspondence of feature points between image pairs is an important first step in many computer vision algorithms that is often assumed, or hand selected. This chapter presents a modular system of robustly computing image pair correspondences from a video sequence by utilizing geometric constraints to guide an iterative process when computing the putative corresponding match set. The method allows for the solution of the correspondence problem in two or more views without human intervention. This chapter makes contributions to the field of computer vision by providing a publicly available robust and accurate method for solving the correspondence problem over 2 and 3 views. Furthermore, it presents a novel methodology to select frames based on the disparity of corresponding features via feature tracking as a precursor to the correspondence problem. Consequently, algorithms that rely on the correspondence problem to be solved apriori can now be solved in an automated fashion. This allows long image sequences to be handled where manual selection would no longer be possible and conditions the input for photogrammetry algorithms which makes the solutions to these photogrammetry problems more tractable. For example, by appropriately spacing the input images, the bundle adjustment is more likely to converge.

Chapter 4 – Video Segmentation

This chapter is a slight departure from the others in that we steer away from multiple view geometry based computer vision and use feature-tracking techniques to segment commercial video clips for use in applications such as content-based video, image indexing and retrieval, video index creation and video database population. There has been much work concentrated on creating shot boundary detection algorithms in recent years. However a truly accurate method of cut detection still eludes researchers in general. Cut detection methods can all be classified based on the various inter-frame differencing schemes that they employ. In this work we present a scheme based on stable feature tracking for inter frame differencing. Furthermore, we present a method to stabilize the differences and automatically detect a global threshold to achieve a high detection

rate. We compare our scheme against other cut detection techniques on a variety of data sources that have been specifically selected because of the difficulties they present for other differencing techniques due to quick motion, many small shots and computer-generated effects. In this study our goal is to improve the accuracy of cut detection, particularly for difficult image sequences. The method improves on both speed and accuracy over existing feature-based video segmentation methods. Our new method also improves accuracy over previously established histogram-based methods. Finally, our new method also allows for annotation of the video clips into categories based on feature motion vectors.

Chapter 5 – Autocalibrating Long Image Sequences

In order to get metric information from an image sequence it is required that we compute the internal camera (intrinsic) parameters. Whilst this has been studied in the past and complex methods exist that require a projective reconstruction to be computed, we examine a simpler method that uses the fundamental matrix and genetic algorithms to estimate the camera parameters with similar accuracy to the complex methods. Finally, in situations such as video processing, the complex methods are not well suited because they do not scale up as the image sequences become large, whereas the method presented is ideally suited for long images sequences. This chapter makes contributions to the field of computer vision and computational video processing by allowing very long image sequences used in an autocalibration step necessary for computer vision tasks such as 3D reconstruction. The method presented improves speed while maintaining accuracy when compared to other complex methods and therefore allows for the autocalibration of video capable cameras.

Chapter 6 – Synchronizing Multiple Video Sequences

At this stage the thesis has examined techniques for performing computation on a single video sequence. In this chapter, we increase the complexity by trying to utilize multiple unsynchronized video streams. In order to make use of any known stereo vision algorithm, we must first synchronize the video streams by identifying the frames that correspond to the exact same moment in time. We contribute to the field by allowing a fast and automated method for temporal synchronization, which is often assumed, of multiple

video cameras in 2D without the requirement of forcing extrinsic calibration, requiring certain motion or scene constraints.

Chapter 7 – Conclusions and Open Problems

Finally, in this chapter we will summarize the work and future research opportunities in this area. The main contribution of this chapter is a list of open and interesting problems.

Chapter 2

Background

The goal of a machine vision system is to create a model of the real world from static images [13]. Using these models, applications can perform some function such as robot navigation or object tracking. Although early work in computer vision systems mainly concerned itself with static scenes, the importance of dynamic scenes, object motion and video input can not be ignored. Research naturally broadened, and the projective vision and computational video branches of the computer vision tree evolved.

2.1 Computational Video

A computational video system uses as input a series of consecutive images, where each image is of a scene at a given point in time. Video data, which is typically anywhere from 10 fps (frames per second) to 30 fps, makes an ideal input source for dynamic vision applications. With the increasing availability of multimedia applications and hardware for capturing video, it is easy to see how projective vision and computational video are complimentary technologies.

A sequence of frames offers a lot more information regarding a scene, but obviously it requires a level of computation that is much higher as well. For example, one such system's aim is to detect changes and subsequently determine the motion of the objects in the scene as well as the camera positions. The relationship between object motion and video camera motion falls into one of the following four models:

1. Stationary Cameras / Stationary Objects
2. Stationary Cameras / Moving Objects
3. Moving Camera / Stationary Objects
4. Moving Cameras / Moving Objects

Each of these categories requires slightly different techniques, but some fundamental concepts, mathematical background and practices are common to all four. A subset of the frames from video sequences form an identical computer vision problem that would

be formed from single frame cameras located at the exact same position of the video camera at the time the frames were taken.

Classical computer vision requires calibrated camera systems, but recent developments in the field have pushed the requirement of calibration out of the way for some problems. Calibrated computations require a transformation matrix known as the essential matrix [7] that characterizes the transformation between two calibrated camera view points. In uncalibrated projective vision, a transformation matrix can be estimated using the two images [10][11]. This estimated matrix is known as the fundamental matrix[8][9], and has the same basic conceptual uses that the essential matrix has. The relationship between the fundamental matrix and the essential matrix becomes clear when the fundamental matrix is calculated for a calibrated system. As expected, the fundamental matrix derived from a calibrated system correctly yeilds the essential matrix of the given system. These ideas are covered in more detail later on in the chapter. In a computational video system, one core challenge is to determine the appropriate model and then apply that correct techniques for that model.

2.2 Camera Models

If we consider the simplest model for a camera, we would be considering the pinhole camera, which is also known as the perspective camera. A pinhole camera consists of two planes, with a minute hole punched in the focal plane to allow rays of light through to fall upon the image plane. The rays of light pass through the pinhole on the focal plane in such a way as to produce an inverted image on the image plane. This simple camera has an optical centre located at C on the focal plane and the image plane is located at distance *f* from the focal plane. Figure 1 shows the pinhole camera model.

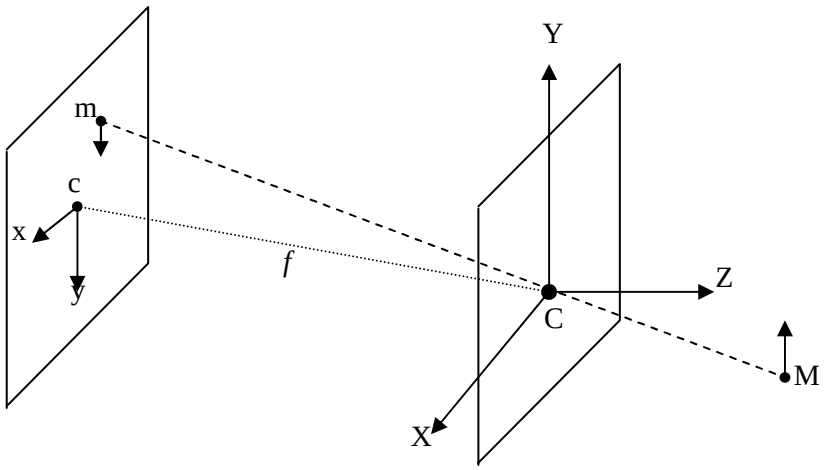


FIGURE 2.1: THE PINHOLE CAMERA MODEL.

Each point **M** in the object forms a straight line through the optical centre C with its corresponding image point **m**. This type of projection of the 3D world to a 2D plane is known as the perspective projection. From a geometric standpoint, it makes no difference if we replace the image plane with a virtual image plane in front the focal plane. Figure 2 below shows a pinhole camera with a virtual image plane.

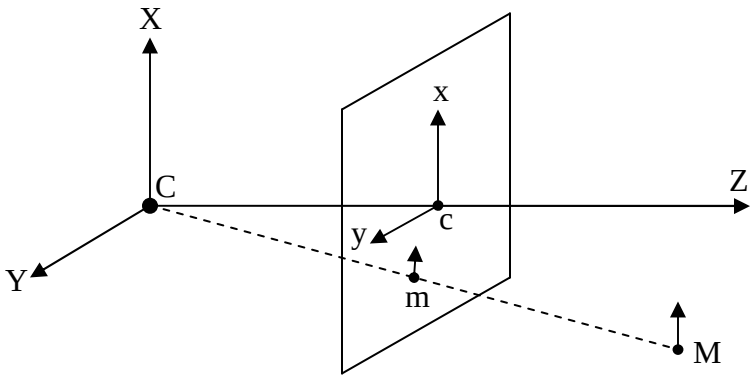


FIGURE 2.2: THE PINHOLE CAMERA WITH A VIRTUAL IMAGE PLANE.

The relationship between 3D and 2D coordinates can be written linearly as[14]:

$$\begin{bmatrix} x \\ y \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.1}$$

where f is the focal length of the camera. The point lies on the image plane at [x/s, y/s] if s > 0. If s < 0, then the point lies behind the image plane and cannot be projected. For the case where s is exactly 0, we have what is termed 'a point at infinity', and the projected points are not defined. The 3x4 matrix above is known as the *perspective projection* matrix and is usually denoted by **P**.

$$\mathbf{P} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.2}$$

Digital cameras, scanners and other capture devices mimic this perspective camera model due to much effort put forward to minimize and eliminate lens distortions. This allows us to use projective geometry to characterize the geometric relationships between the images and the real world. We continue with a brief overview of projective geometry.

2.4 Overview of Projective Geometry

When we think of the world and space around us, we think of Euclidean space; so it would seem logical that machine vision would work with Euclidean geometry. This is not the case, and a more generalized form of geometry is used. Euclidean geometry is a special case of projective geometry, and questions are often more easily answered in the more general context of projective geometry [15]. Not only does this make our computations simpler, but it is also ideal as we are working with images that are merely projections of Euclidean coordinates to a plane. Any projection of a point (in the Euclidean coordinate system) $M_w = [X_w, Y_w, Z_w]^T$ to the image plane $m = [x,y,s]^T$ can be described using simple linear algebra.

$$m = \mathbf{P}M \tag{2.3}$$

where m is the projected homogeneous coordinates of the point M in the image.

Any projection of a point (in the world coordinate system) $M_w = [X_w, Y_w, Z_w]^T$ to the image plane $m = [x,y]^T$ can be described using simple linear algebra.

$$sm = \mathbf{P}M \tag{2.4}$$

2.4.1 Projective n-spaces

Any point p, of an n dimensional projective space $\mathbf{P^n}$, is represented by a vector of n+1 elements not all zero. The elements of this vector are commonly referred to as projective coordinates or homogeneous coordinates. Any two vectors $\mathbf{x} = [x_1,\ldots x_{n+1}]^T$ and $\mathbf{y} = [y_1,\ldots y_{n+1}]^T$ are considered equal (representing the same point) if and only if there exists a scalar $\lambda \neq 0$ such that $x_i = \lambda y_i$ for all elements in the vector.

When we do use homogeneous coordinates, the algebra for projective geometry becomes very simple. For example [16], the Cartesian coordinates of the point where two lines $ax + by + c = 0$ and $rx + sy + t = 0$ intersect is:

$$(bt - cs , cr - at) / as - br \tag{2.5}$$

Whereas in homogeneous coordinates, the intersection of $[a,b,c]^T$ with $[r,s,t]^T$ is

$$[bt - cs, cr - at, as - br]^T \tag{2.6}$$

which we easily recognize as the vector (cross) product. While this example is in projective 2 space, its truth exists throughout the dimensions. Notice that we not only made the algebra simpler, but we removed the division operation, which is costly on a computer.

2.4.2 Collineations

Projective transformations (collineations) are linear transformations. In other words, it maps features in one projective space to the same features in the same projective space. These transformations are characterized by a (n+1)×(n+1) non-singular matrix **A** so that $\lambda p_2 = \mathbf{A}p_1$. The matrix **A** has the following mapping properties:

- Collinear features remain collinear
- Concurrent features remain concurrent
- Incidence is preserved

It is rather easy to see that the set of collineations transforming $\mathbf{P^n}$ onto itself form an algebraic group. This group is known as the *projective group*.

Theorem: *For any two linearly independent sets of points in projective n space that forms a basis; say* $B = \{b_1,\ldots,b_{n+2}\}$ *and* $C = \{c_1,\ldots,c_{n+2}\}$*, there exists a collineation* $\mathbf{A}$ *such that* $\delta B_i = \mathbf{A}c_i$.

We borrow this **proof** from [15]:

We can choose a matrix **P** and a set of nonzero scalars $\alpha_1,\ldots,\alpha_{n+2}$ such that:

$$\mathbf{P}e_i = \lambda_i b_i$$

Where e_i is the projective basis. Similarly, we chose **Q** and $\mu_1,\ldots,\mu_{n+2}$

$$\mathbf{Q}e_i = \mu_i c_i$$

Then

$$\mathbf{PQ}^{-1}b_i = (\mu_i/\lambda_i)c_i$$

Thus we let $\delta_i = \mu_i/\lambda_i$ and $\mathbf{A} = \mathbf{PQ}^{-1}$

QED

2.4.3 The Projective Plane (P^2)

The projective plane is important to us, as it forms the basis of our work. Sensors that produce 2 dimensional projections of the 3D world are common. In fact, the image plane of a typical CCD camera is simply modeled as a 2D projective plane. There are 4 basic structures in the projective plane that we need to be concerned with; these are points, lines, pencils, and conics.

2.4.3.1 Points and Lines

From the previous discussion, we know that points in $\mathbf{P}^2$ are represented by a vector with three elements (m_1,m_2,m_3). Other than points, we have lines which are also defined by three numbers, not all zero. The *principle of duality* states that *lines* and *points* are represented the same way in two-dimensional projective geometry. That is, a projective point m is given as $[m_1,m_2,m_3]^T$ and a projective line l is given as $[l_1,l_2,l_3]^T$. The point p comes from the perspective projection of three-space down to two (see below), and the line equation is simply:

$$L_1m_1 + l_2m_2 + l_3m_3 = 0. \qquad (2.7)$$

Due to this principle, we can use the terms point and line interchangeably. This is when we discuss lines, we are implicitly discussing points in the exact same manner.

2.4.3.2 Lines at Infinity

Of all the possible lines in $\mathbf{P}^2$ a special subset exists when the third element of the line in homogeneous coordinates is equal to zero. These lines are known as *lines at infinity* usually denoted by ℓ_∞. The important implication of all this is that in projective ge-

ometry, any two distinct lines (even if they are parallel) will always intersect. Parallel lines happen to intersect at the point at infinity!

2.4.3.3 Pencils of Lines

Pencils of lines have numerous applications in vision, especially in stereo and motion. Pencils are the set of all lines in $\mathbf{P^2}$ that pass through a fixed point. The set of epipolar lines for a given fundamental/essential matrix are a pencil because they all pass through the epipole.

2.4.3.4 Conics

Conics are a set of points on the projective plane that satisfy the equation:

$$S(x) = x^T A x = 0 \tag{2.8}$$

Where A is a 3x3 symmetric matrix. The equation defines the conic up to a scale factor, and is dependent on 5 parameters.

2.4.4 The Projective Space $\mathbf{P^3}$

In projective 3 space, points are represented by a vector with four elements. Recalling the principle of duality, we see that the dual entity to a point $\mathbf{P^3}$ is a plane. Similarly to $\mathbf{P^2}$ a point $M = [M_1, M_2, M_3, M_4]$ is contained in a plane $\mathbf{\Pi}$ if and only if:

$$\mathbf{\Pi}^T M = 0 \tag{2.9}$$

Lines in $\mathbf{P^3}$ are simply the intersection of two planes, and thus can also be expressed as a linear combination of two points. i.e. $L = \lambda_a M_a + \lambda_b M_b$

2.4.4.1 Quadrics

Quadrics are the 3-space equivilant of conics in 2-space are a set of points on the projective plane that satisfy the equation:

$$\Pi^T Q \Pi = 0 \tag{2.10}$$

Where Q is a 4x4 symmetric matrix. The equation defines the conic up to a scale factor, and is dependent on 9 parameters.

2.4.5 Strata of Geometries

Our world is a 3D Euclidean space. When we are dealing with images, we are in the simpler structure of projective geometry, between these two spaces lie two intermediate geometries, affine and metric. Thus the order of strata from simplest to most complex is: projective, affine, metric, and Euclidean. The strata are defined by their group of collineations and the features that are left unchanged (invariant). Each strata contains group of transformations that maintain a set of invariant properties. Also, it should be noted that each group is a subgroup of the simpler structure. This means that the metric group is a subgroup of the affine group, and both are a subgroup of the projective group. The invariant properties of a geometric strata are not changed during a transformation that belongs to the same geometry. Knowing the invariant properties and being able to recover them allow us to change strati. Often we wish to upgrade to a higher level in the strata, in fact this is what we are doing when we go from a sequence of images to a 3D Euclidean model. In the following sections, we detail each of the geometric strata in 3D space, their group of transformations and invariants. 3D space is chosen as it is relevant in going from an image sequence to a 3D model. We simply reconstruct the projection matrix and change strata to metric or optimally Euclidean.

2.4.5.1 Projective Strata

The least structured of all strata is the projective stratum. By this, we mean that the projective stratum has the least number of invariants and the largest group of transformations. In 3D space, the projective transformation matrix is a 4x4 invertible matrix.

The invariant property of the projective strata is cross ratio. The cross ratio is defined as follows: Given any four collinear points M_1, M_2, M_3, and M_4; and their respective projective parameters α_1, α_2, α_3, and α_4. The cross ratio is defined as:

$$\{M_1,M_2; M_3, M_4\} = \frac{\alpha_1 - \alpha_3}{\alpha_1 - \alpha_4} : \frac{\alpha_2 - \alpha_3}{\alpha_2 - \alpha_4} \tag{2.11}$$

The cross ratio extends simply to higher dimensions. In the case of $\mathbf{P}^2$ in the figure below, we see how the cross ratio of four lines is defined as the cross ratio of points that intersect with another line. Obviously we can do a similar operation with planes.

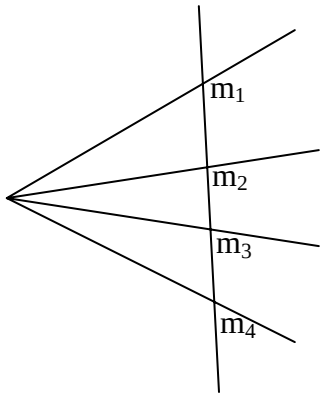


FIGURE 2.3: CROSS RATIO OF 4 LINES IN P^2

2.4.5.2 Affine Strata

The next least structured of all strata is the affine stratum. In 3D space, the affine transformation matrix is also a 4x4 matrix. The affine strata adds parallelism , relative distances, and the plane/line at infinity as invariants. See table 2 below for complete details on the form of the affine transformation matrix.

2.4.5.3 Metric Strata

The metric stratum is also commonly referred to as the group of similarity transformations. This group corresponds to the Euclidean group, but only to a scale factor. This is the highest level of geometry we can reach without knowing some measurement of distance between points. As with all 3D transformations, we represent a metric transformation with a 4x4 matrix when we use homogenous coordinates.

Due to its close relation to the Euclidean group of transformations, we know that we have 3 degrees of freedom from the orthonormal rotation matrix and 3 degrees of freedom for the translational aspects in X, Y and Z. When we add one more degree of freedom due to the scale, we end up with a total of seven degress of freedom. The two new invariants are relative length and angles. Most importantly, at this strata, transformations leave the plane at infinity unchanged and transforms a conic to a conic of the exact form. This fact is useful in autocalibration techniques, as it helps in estimating the intrinsic camera parameters.

2.4.5.4 Euclidean Strata

The Euclidean stratum is the one most familiar to us. Having six degrees of freedom, 3 rotational and 3 translational; the invariants are identical to the metric strata without having a scale invariant.

2.4.5.5 Strata Review

In table 2 below, the properties of the various strata are reviewed. In figure 6 below, the visualization of a cube throughout the different strata is shown[17].

GEOMETRIC STRATUM	DEGREES OF FREEDOM	TRANSFORMATION MATRIX (3-SPACE)	INVARIANTS
Projective	15	$\begin{bmatrix} p_{11} & p_{11} & p_{11} & p_{11} \\ p_{11} & p_{11} & p_{11} & p_{11} \\ p_{11} & p_{11} & p_{11} & p_{11} \\ p_{11} & p_{11} & p_{11} & p_{11} \end{bmatrix}$	Cross ratio
Affine	12	$\begin{bmatrix} p_{11} & p_{11} & p_{11} & p_{11} \\ p_{11} & p_{11} & p_{11} & p_{11} \\ p_{11} & p_{11} & p_{11} & p_{11} \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Relative distance, Parallelism, Plane at infinity
Metric	7	$\begin{bmatrix} \lambda r_{11} & \lambda r_{12} & \lambda r_{13} & t_x \\ \lambda r_{21} & \lambda r_{22} & \lambda r_{23} & t_y \\ \lambda r_{31} & \lambda r_{32} & \lambda r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Relative distances, Angles, Absolute conics
Euclidean	6	$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Absolute distances

Table 2: Information regarding the different strata.

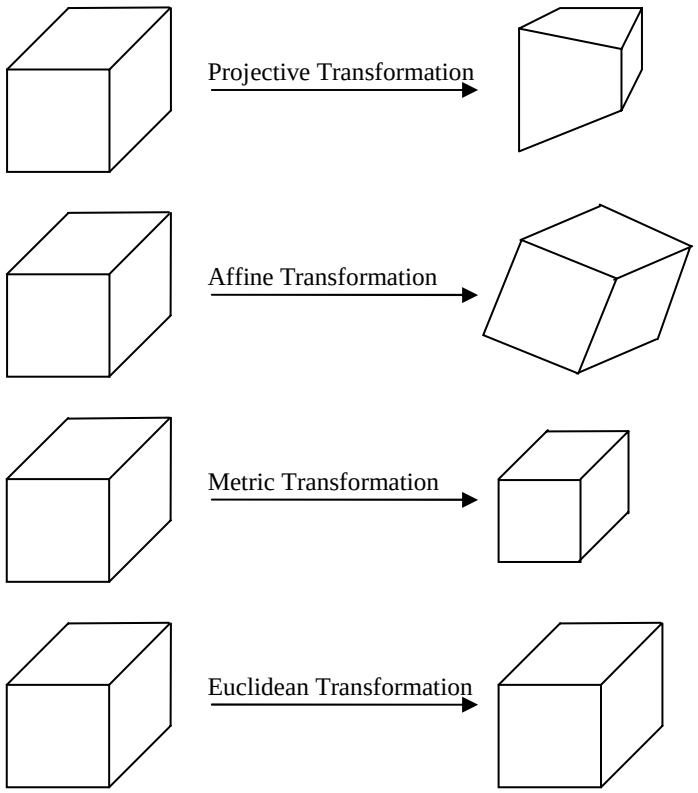


FIGURE 2.4: SHAPE DISTORTIONS FOR EACH GROUP OF TRANSFORMATIONS IN 3D.

2.4.5.6 Changing Strata

When we are going to change strata, it is important to realize that we are in fact upgrading our geometric representation to one that has a stronger structure. It is obvious that starting with images, we are in the projective strata, and that if our goal is model building, we would ideally like to progress through to the Euclidean strata. Thus our ideal set of changes will take us from projective to affine, affine to metric, and metric to Euclidean.

2.4.5.6.1 Projective to Affine

In order to upgrade to affine from the projective strata, one must first locate the plane at infinity. This task can be done if some affine properties of the scene are known. Parallel lines or planes intersect at infinity. For example, it the scene contains a building, one can effectively consider the corners of the building simply as the corners of a cube. The pairs of edges of this cube that are effectively parallel, will intersect at the plane at infinity. As long as no three points are collinear, we have found the plane at infinity.

With these three points, we can use the invariant property of cross ratio to compute the point at infinity M_∞. I.e. M_0,M_1,M_2 are our points that define the plane, the cross ratio is $\{M_0: M_1; M_2:M_\infty\}$

Once the plane at infinity has been successfully identified, the upgrade to affine consists only of bringing the plane at infinity to it canonical position $[0,0,0,1]^T$. The straightforward approach would be:

$$T_{PA} = \begin{bmatrix} I_{3x3} & 0_3 \\ \Pi_\infty & 1 \end{bmatrix} \tag{2.12}$$

Where the bottom row is actually the normalized plane at infinity.

2.4.5.6.2 Affine to Metric

Going from affine (or projective for that matter) to metric requires us to find the absolute conic. Because we are already in the affine space, we know the location of the plane at infinity. This is important because we know that the conic is located in this plane.

Once the conic has been identified, we only need to bring it to its canonical form in the metric strata. On possible choice for the upgrade is[Pol99]:

$$T_{AM} = \begin{bmatrix} A-1 & 0_3 \\ 0_3 & 0 \end{bmatrix} \tag{2.13}$$

Thus going form projective to metric is simply

$$T_{PM} = T_{AM}\, T_{PA} \tag{2.14}$$

2.4.5.6.3 Metric to Euclidean

In order to go from metric to Euclidean, we need to have some actual measurements so that we can determine the scale factor that the metric strata is at. Once this scalar is computed, it is simply a matter of applying that scale factor to the strata.

2.5 Camera Calibration

In many applications a change in coordinate systems is inevitable, and most likely occurs quite often. The purpose of calibration is to determine the relationship between coordinate systems. Camera calibration falls into two logical subsets, intrinsic and extrinsic. Intrinsic calibration concerns itself with the internal geometry of a physical camera, while extrinsic calibration deals with the external properties of the camera such as position.

If we have knowledge of the intrinsic parameters, we are able to perform metric measurements with the camera. If we do not have the intrinsic parameters we have what is termed an uncalibrated camera, and cannot get exact metric measurements. We can however do many things with uncalibrated cameras including reconstruction, motion detection, and possibly autocalibrate the camera itself.

2.5.1 Intrinsic

Intrinsic parameters relate the change in coordinate systems from image coordinates to pixel coordinates. The main goal of intrinsic calibration is to rectify errors in the manufacturing of the capture device. In practical applications that use physical cameras, the intrinsic parameters are very important for several reasons:

- Typical cameras (such as CCD's) have varying pixel coordinate systems that are not necessarily the same as the projection coordinate system.
- Manufacturing defects cause the axes of capture devices to be at angles other than 90 degrees.
- The projection planes' origin may not coincide with the optical axis of the capture device due to lens distortion or other effects. i.e. The pixel grid is not orthogonal with the optical axis.

If we examine Figure 2.5 below, we see how non-orthogonal axes cause the need for a translation to a different but more accurate coordinate system.

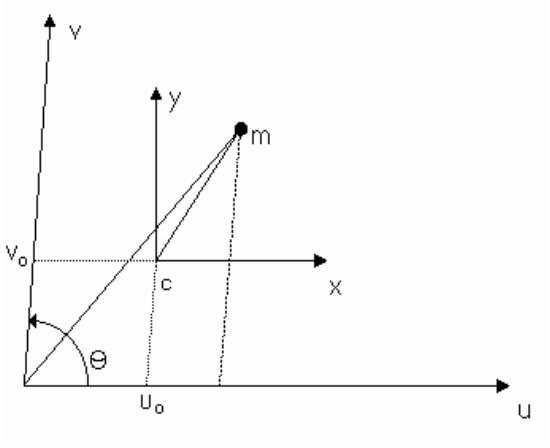


FIGURE 2.5: INTRINSIC PARAMETERS

The intrinsic parameters are the skewness (θ) or how rectilinear the pixels really are. The optical centre (Uo,Vo). The aspect ratio (α), which describes the ratio of the width to the height of a pixel. The calibration matrix **K** is:

$$K = \begin{bmatrix} \alpha x & \theta & Uo \\ 0 & \alpha y & Vo \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

This transformation from image to pixel coordinates is also linear and can be written as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = [K] \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.16}$$

With an image being formed by a perspective projection, followed by intrinsic calibrations gives us, our projective transformation matrix becomes:

$$\mathbf{P}_{\text{new}} = \mathbf{K}\mathbf{P}_{\text{old}} \tag{2.17}$$

2.5.2 Extrinsic Calibration

The main goal of extrinsic calibration is to take coordinates in the world coordinate system and transform them to the camera coordinate system. Extrinsic parameters are commonly used to change views and move through virtual camera views. To go from one system of coordinates to another, we require a rotation **R** and a translation **t** as seen in Figure 2.6. The relationship between the world and the camera coordinates are

$$M_c = \mathbf{R}M_w + \mathbf{t} \tag{2.18}$$

Where **R** is a 3x3 rotation matrix and **t** is the translation vector. Figure 2.6 below shows the transformation between coordinate systems.

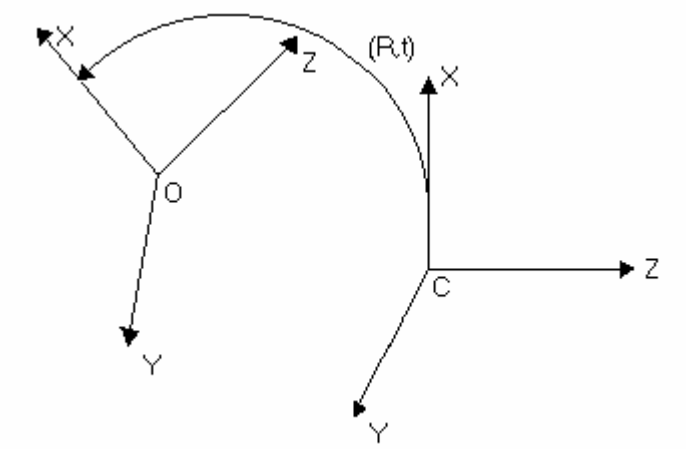


Figure 2.6: EXTRINSIC PARAMETERS FOR A GENERIC CAMERA IN THE WORLD

For any given point M in the original coordinate system, the new point M^1 can be linearly calculated using

$$M^1 = \mathbf{D}M \tag{2.19}$$

where

$$\mathbf{D} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.20}$$

From equation 2.4, we simply add the transformation D

$$sm = \mathbf{PD}M \tag{2.21}$$

There are six extrinsic parameters which are the rotational angles between axes and the translation along the 3 axes.

2.6 Planer Transformations

When we are dealing with co-planer points (i.e. All points $M = [X,Y,Z,1]^T$ where Z is equal) in the world coordinate system, we simply have to compute planer transfor-

mations. The planer transformations are a special case of the 3D transformations known as *homographies.* If we choose a world coordinate system such that the plane of the points has a zero Z value, the projection matrix **P** which is normally 3x4, reduces to a 3x3 matrix that defines a general plane to plane transformation [19].

$$\mathbf{H} = \mathbf{P}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{2.22}$$

Where **H** is simply **P** with the 3rd column zeroed out and can be effectively ignored yielding

$$\mathbf{H} = \begin{bmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.23}$$

Where p_{ij} is the value from **P** in row i and column j.

To illustrate how the homographies work in relation to regular 3D transformations, the example below [19] shows how homographies work only on co-planer points. In Figure 2.7 below, we see the same scene taken from two different camera viewpoints.

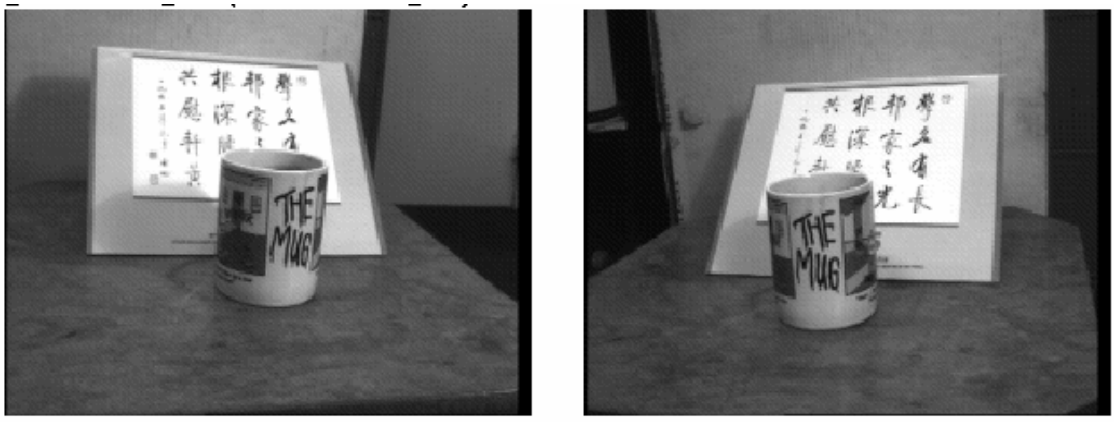


Figure 2.7: TWO VIEWS OF THE SAME SCENE

If we use a homography to translate from one view point to the other viewpoint, and we use the paper with the Chinese text as our planer points, we get the image in Figure 2.8.

Figure 2.8: TRANSFORMATION OF LEFT IMAGE TO THE VIEW POINT OF THE RIGHT IMAGE

The warped image in Figure 2.8 appears to be a complete image, but if we examine the mug closely, we notice that is it not quite right. To show how planer objects were correctly transformed while other points not in the plane are warped, we superimpose the real camera view with the translated camera view to get Figure 2.9 below.

Figure 2.9: RIGHT VIEW FROM FIG. 2.4 SUPERIMPOSED ONTO FIG 2.5

What use does such a transformation have when it is so evidently erroneous? Simply put, synthetic view points of co-planer points can be generated. The error from the above example resulted from the points that were not co-planer, i.e. the mug. The co-planer points on the sheet of paper were perfectly transformed. This allows us to easily create mosaic images that can be viewed from different angles. Large scenes that contain co-planer surfaces can be rendered as one single image that can be viewed from many different angles. While multiple single images may be required to capture a scene, it may

be viewed as a single image. Consequently, to render a large image we need only minimally overlapping planar input images.

2.7 Stereo Vision

The goal of stereo vision is to generate some depth information about a scene. Using the disparity information from two or more images, it is possible to calculate depth values for a calibrated system. Disparity is the distance between two points in an object found in both images. It is easy to see how disparity is involved in depth calculations with the following simple demonstration. Find an object in your field of view that is between 2 and 3 meters away, and look at it. Hold your index finger out at arms length and look at the scene with your left eye, then with your right. Notice how the closer object (your index finger) 'moves' more than the farther object. This separation of the object in your left and right views is known as the disparity. Objects that are closer to the cameras, in this case your eyes, have a larger disparity than do objects that are farther away. The selected points used for calculation of the disparity are known as conjugate or corresponding pairs. Automated selection and verification of conjugate pairs is known as the *correspondence problem*. The geometry that relates conjugate pairs is known as the *epipolar geometry* and exists between any two camera systems [10].

2.7.1 Correspondence Problem

It has been implied that stereo vision requires the selection and searching for points to be used as conjugate pairs. What we have failed to mention is that the detection of conjugate pairs in two images is extremely challenging. This area of research is known as the correspondence problem and is covered extensively in Chapter 3.

2.7.2 Epipolar Geometry

The simplest form of stereo vision involves a pair of cameras with a fixed x displacement [13]. In Figure 2.10 below, we see that the point M in the scene is in both the left and the right image planes as point m. The plane that passes through the point M and both camera centres is the epipolar plane. The epipolar lines are defined as the intersection of the epipolar plane and image plane. The epipoles are defined as the point where all epipolar lines intersect. As well, the epipole is defined as the intersection of the image

plane with a line between the optical centres of two cameras. Figure 2.7 below illustrates such an epipolar geometry.

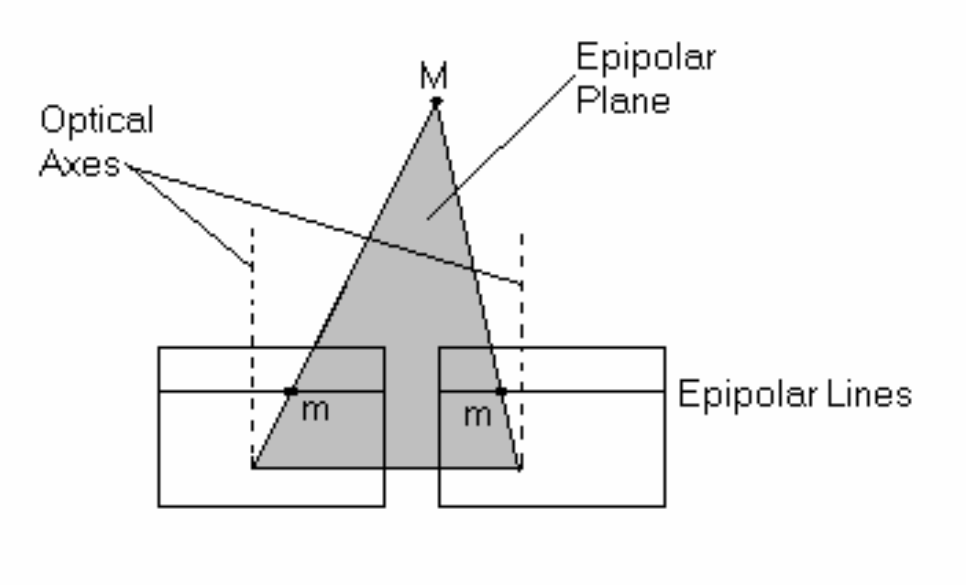


FIGURE 2.10: EPIPOLAR GEOMETRY OF TWO CAMERAS WITH FIXED X DISPARITY

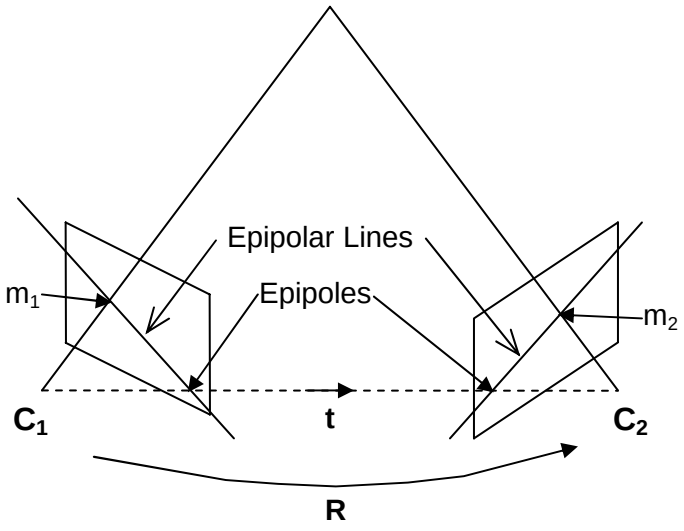


Figure 2.11: EPIPOLAR GEOMETRY OF TWO CAMERAS IN AN ARBITRARY POSITION

The fundamental and essential matrices offer a great computational advantage in matching a point in one image with the same point in another image. Since the corresponding points must lie on the epipolar line, our search for corresponding points has

been reduced from 2 dimensions down to 1. We now only have to search along the epipolar line to find the corresponding point. This is known as the *epipolar constraint* [10].

The epipolar geometry of two stereo images is related by a simple transformation characterized by the *epipolar equation*. First introduced by Longuet-Higgens [7] in 1981, the epipolar equation produces the Essential Matrix. The Fundamental Matrix characterizes the epipolar geometry between two uncalibrated cameras.

2.7.3 Essential Matrix

Calibrated stereo vision gives us the ability to calculate depth metrics of the scene. These Z values allow applications such as robotic vision and navigation to be possible. In calibrated stereo vision we are working with the *essential matrix* **E**, which is completely encompassed by the rotation and translation between the two cameras. Because the cameras have a known calibration, we can work in the normalized image coordinate system.

The relationship between two points in two separate images can be described mathematically as

$$m_1^T \mathbf{E}\, m_2 = 0 \tag{2.24}$$

where

$$\mathbf{E} = \mathbf{t} \times \mathbf{R}. \tag{2.25}$$

And this is referred to as the *epipolar equation.*

2.7.4 Fundamental Matrix

When we are given two uncalibrated cameras **K,** the calibration matrix, is unknown to us and we therefore cannot use **K** to easily transform the pixel coordinates into normalized image coordinates. This forces us to work in the pixel coordinate system, and the epipolar geometry is still characterized by the epipolar equation.

$$m_1^T \mathbf{F}\, m_2 = 0 \tag{2.26}$$

where points m_1 and m_2 are corresponding points in image 1 and image 2 respectively

F can be characterized in terms of the essential matrix and the camera calibration matrices

$$\mathbf{F} = \mathbf{K}_1^{-T}\mathbf{E}\mathbf{K}_2^{-1} \tag{2.27}$$

Theorem 1([8] [9]) – *For any two views I_1 and I_2 of an uncalibrated scene, there exists a fundamental matrix of rank 2 that adheres to the following property: For all corresponding homogeneous points (m_1, m_2) in I_1 and I_2*

$$m_1^{\,T}\,\mathbf{F}\,m_2 = 0 \tag{2.28}$$

Proof

Let M represent a real point in 3D space. M = (x,y,z). Also, let m_1 and m_2 be the homogeneous coordinates of the image points in image 1 and 2 respectively. Assuming the initial point m_1 is at **0** and the corresponding point m_2 is at **t**. The unknown intrinsic camera parameters **K** make the camera transformation matrices to be

$$[\mathbf{K}|\mathbf{0}] \text{ and } [[\mathbf{K}][\mathbf{R}]|[\mathbf{K}]\mathbf{t}]$$

before and after the motion **t** respectively. Determining the epipolar line in the second image for the point x, the camera centre and point at infinity become

$$[\mathbf{K}]\mathbf{t} \textbf{ and } [\mathbf{K}][\mathbf{R}][\mathbf{K}]^{-1}\mathrm{m} \text{ respectively.}$$

As a result, the epipolar line is given by

$$l = [\mathbf{K}]\mathbf{t} \times [\mathbf{K}][\mathbf{R}][\mathbf{K}]^{-1}\mathrm{m}$$

Since m' lies on the epipolar line l

$$m'^{\,T}\,\mathbf{F}m = 0$$

thus

$$\mathbf{F} = [\mathbf{K}]\mathbf{t} \times [\mathbf{K}][\mathbf{R}][\mathbf{K}]^{-1}$$

Which is

$$\mathbf{F} = [\mathbf{K}]^*[\mathbf{t} \times \mathbf{R}][\mathbf{K}]^{-1}$$

Where [**K**]* is the adjoint of K, and from (2.25)

$$\mathbf{F} = \mathbf{K}_1^{-T}\mathbf{E}\mathbf{K}_2^{-1}$$

Which is (2.27) and known to exist.

QED

From the discussion above, it should be clear that epipolar geometry depends upon the orientation and internal physical characteristics of two cameras. The geometry does not depend on the structure of the scene. i.e. The 3D points external to the cameras have no bearing on the actual geometry.

2.8 Robust Methods for Computing the Epipolar Geometry

The above methods make the assumption that there is no noise present in the corresponding pairs, which in practical situations is simply unreasonable. In [20] it was shown that mismatches and noise are unavoidable in practical situations and robust methods are required to estimate the epipolar geometry. The generation of the correspondence pairs results in two potential types of errors, these are: incorrect location of a pair (inlier error) match, and incorrect pairing (outlier error). Figure 2.12 below shows these two types of errors.

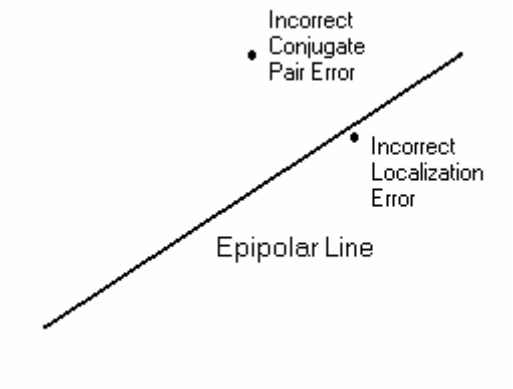


Figure 2.12: THE TWO TYPES OF CONJUGATE PAIR ERRORS

Inlier error is assumed to exhibit a Gaussian distribution. This means that most error will be small and within one or two pixels. However, a few points will be incorrectly localized with an error of more than 3 pixels. Error of more than 3 pixels will severely degrade the estimation of the fundamental matrix.

Incorrect correspondence (conjugate) pairing is a more serious problem that occurs when two points that are not a valid correspondence are selected incorrectly as being a valid pair. Because the epipolar constraint may not yet available, the search for conjugate pairs must be conducted in 2D. Thus, the potential number of incorrect conjugate pairs

could be quite high. These invalid conjugate pairs completely spoil the estimation of the fundamental matrix and therefore **F** would be inaccurate.

The precision of the fundamental matrix is seriously affected by errors of these types, and robust methods need to be employed when calculating **F**. Robust methods try to minimize the error caused by inliers, and remove the inaccuracy caused by outliers. Since one outlier will make the estimated fundamental matrix **F** useless, we need to use as many sets of valid pairs as we can to compute **F**. Several classes of robust techniques exist to compute the fundamental matrix, these are highlighted very well in [11] and [21]. The algorithm that performed best was RANSAC (RANdom Sample Concensus).

2.8.1 RANSAC

RANSAC, performs an estimation by randomly selecting the minimum required number of correspondences to compute **F**. For each **F** computed, the set of inliers is calculated. The **F** with the highest consensus of inliers is selected to be the final computed **F** i.e. the one with the most support**.** This computation is repeatedly completed until a certain level a certainty is achieved.

The basic random sampling algorithm is as follows:

Repeat for M samples

- *Select a random sample of the minimum required correspondences to estimate a valid fundamental matrix F.*
- *Compute a putative matrix F*
- *For all putative correspondences, compute the set of inliers*
- *Select the F with the greatest number of inliers over all samples*

2.9 Three View Geometry

The next natural step from epipolar geometry is to add a third camera view. The trifocal tensor approach is one such extension and maintains its basis in projective geometry. This model has been proposed and developed by Hartley [18], Sashua [22], Torr [23], and Faugeras [24] among others. Figure 2.13 represents the 3-view imaging scenario.

In the 3-view situation, the trifocal plane is formed by the three optic centers C', C'' and C'''. The intersection of this plane with the three image planes produces three lines called the trifocal lines (not shown in figure 8). One could use standard epipolar geometry and consider three fundamental matrices (one for each pair of optical centres) F_{12}, F_{23} and F_{31}. Intersecting epipolar lines should show the position of the point (shown in figure 2.13). However, if a point M is in the trifocal plane, or the optical centres C'.C'', and C''' are collinear, the fundamental matrices cannot determine if its 3 images a point belong to a single 3D point because the epipolar lines are collinear and therefore intersect at more than one point.

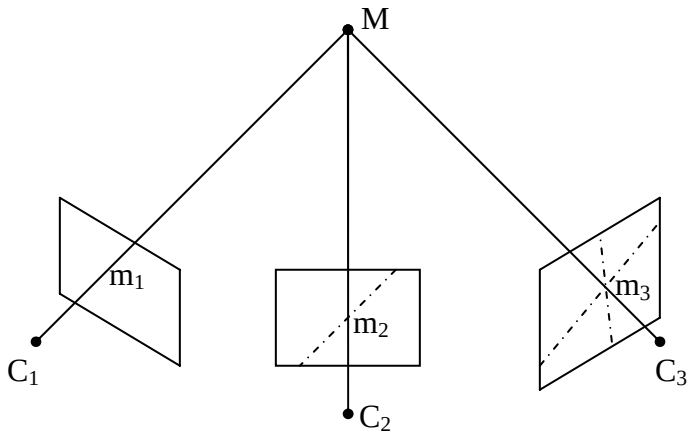


FIGURE 2.13: TRIFOCAL GEOMETRY

In the case of two views, given a point in one image, it is possible to construct a line in another using the fundamental matrix. However, given a point in the first image and a point in the second image, one can directly compute the coordinates of the corresponding third point using a structure called the trifocal tensor which is the analog of the fundamental matrix for 3 view situations.

2.9.1 The Trifocal Tensor

The trifocal tensor is intended to describe line correspondences. This has been a well-known problem to those in the computer vision community dealing with structure from motion [25] [26]. Several years had passed before the tensor was formally identified and defined by Hartley and Shashua [18] [22]. In the projective sense, the tensor is known as the trilinear tensor.

The tensor can be considered as a 3 x 3 x 3 cube operator, defined by 27 parameters in total. Typically, one uses this tensor ($\Im$) to map a line in image 1 (l_1) and a line in image 2 (l_2) to a line in image 3 (l_3). This is known as transfer. This mapping is a linear expression

$$l_3 = \Im(l_1, l_2) \tag{2.29}$$

which is more formally represented by:

$$l_i = \sum_{j=1}^{3}\sum_{k=1}^{3} l_j'' l_k''' \Im_{ijk} \tag{2.30}$$

The tensor can also map corresponding points in two views to their triple corresponding point in the third view. Points are transferred via the following formula:

$$x_l''' = x_i'' \sum_{k=1}^{3} x_k \Im_{kjl} - x_j'' \sum_{k=1}^{3} x_k \Im_{kil} \tag{2.31}$$

The same tensor can be used to transfer lines and points due to the principle of duality.

2.9.2 Constraints on the Tensor

For every three views of a static scene, there exists a 3 x 3 x 3 tensor with the following properties given any three corresponding image points (**m**,**m**',**m**''). For every line **l**' through **m**' and **l**'' through **m**'' in their appropriate views, the trifocal constraint is described by one equation:

$$\mathbf{l}'^{T}[\Im\mathbf{m}]\mathbf{l}'' = 0 \tag{2.32}$$

where

$$[\Im\mathbf{m}]_{ij} = \Im_{1ij}x + \Im_{2ij}y + \Im_{3ij} \tag{2.33}$$

It is important to realize that we are not restricted to using lines with the tensor. In fact, the tensor constraints exist for points as well. If we assume we have a triple correspondence (**u**, **u**', **u**'') the trifocal constraints are defined using four equations:

$$\begin{aligned}
&\mathbf{u}''\Im_{i13}\mathbf{u}_i - \mathbf{u}''\mathbf{u}'\Im_{i33}\mathbf{u}_i - \mathbf{u}'\Im_{i31}\mathbf{u}_i - \Im_{i11}\mathbf{u}_i = 0\\
&\mathbf{u}''\Im_{i13}\mathbf{u}_i - \mathbf{u}''\mathbf{u}'\Im_{i33}\mathbf{u}_i - \mathbf{u}'\Im_{i32}\mathbf{u}_i - \Im_{i12}\mathbf{u}_i = 0\\
&\mathbf{u}''\Im_{i23}\mathbf{u}_i - \mathbf{u}''\mathbf{u}'\Im_{i33}\mathbf{u}_i - \mathbf{u}'\Im_{i31}\mathbf{u}_i - \Im_{i21}\mathbf{u}_i = 0\\
&\mathbf{u}''\Im_{i23}\mathbf{u}_i - \mathbf{u}''\mathbf{u}'\Im_{i33}\mathbf{u}_i - \mathbf{u}'\Im_{i32}\mathbf{u}_i - \Im_{i22}\mathbf{u}_i = 0
\end{aligned}$$

Other combinations of image features also allow us to compute the tensor. The table below shows the feature combinations and the number of equations that describe the trilinear constraints.

Image Features	# of equations
3 points	4
2 points, 1 line	2
1 point, 2 lines	1
3 lines	2

Table 2.3: Overview of trifocal constraints and resulting number of equations

Since we are concerned mainly with points in images, further derivation of the constraints is not necessary to understanding this thesis. A complete set of derivations of the trilinear constraints for points and lines can be found in [30].

2.9.3 Robust Computation of the Tensor

In a manner similar to robustly computing the fundamental matrix, we can use the RANSAC paradigm to compute the tensor.

The basic random sampling algorithm is as follows:

Repeat for M samples

- *Select a random sample of the minimum required triple correspondences to estimate a valid tensor.*
- *Compute a putative tensor T.*
- *For all putative correspondence triplets, compute the set of inliers*
- *Select the T with the greatest number of inliers over all samples*

2.10 N-View Geometry

The next obvious question that comes to mind when considering multiple view geometry is what new constraints if any are found in four views. Triggs [27] and Hartley [28] have examined the concept of the quadrifocal tensor. Triggs claimed the existence of only 3 types of geometric relationships, bilinear (epipolar), trilinear, and quadrilinear linking two, three, and four views respectively. In [29], Faugeras showed that the quadrilinear constraint is a natural result due to the epipolar and trilinear constraints. In fact, as the number of views increases, additional constraints can be expressed using epipolar and trilinear constraints. A complete and formal review of multiple view geometric relationships is given in [30].

2.11 Feature Tracking

Shi and Tomasi [50] present a feature tracking algorithm based on Kanade and Lucas registration technique [49] that selects features which are optimal for tracking in the sense that the tracking equations dictate what characterizes a good feature track. The basic principle of the tracker is that a good feature is one that can be tracked well, so tracking should not be separated from feature extraction. In other words, features are selected because they are optimal for the feature tracking equations, rather then developing a tracking equation for certain features. A good feature is a textured patch with high intensity variation in both x and y directions, such as a corner. We will re-examine the feature selection process after we present the tracking equations. Briefly, features are located by examining the minimum eigenvalue of a 2x2 image gradient matrix. The features are tracked using a Newton-Raphson method of minimizing the difference between the two windows around the feature points.

2.11.1 The Feature Tracking Equations

The tracking algorithm defines a measure of dissimilarity that quantifies the change in appearance of a window around a feature in the first frame and the current frame. The algorithm allows for affine distortion changes in the window. However, a pure translation model of the motion is used to track the selected best features through the sequence. For reliable and fast processing, the maximum displacement is limited, but larger than that of conventional optical flow approaches. Feature tracking is performed on the luminance channel (grey map) for the video frames. The luminance channel is computed as follows:

$$\text{Luminance} = \text{Red}*0.299 + \text{Green}*0.587 + \text{Blue}*0.114 \tag{2.34}$$

Given a point p in an image I, and its corresponding point q in an image J, the displacement vector δ between p and q is best described using an affine motion field:

$$\delta = Dp + t \tag{2.35}$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \tag{2.36}$$

is a deformation matrix and t is the translation vector of the centre point of the tracked feature window. The translation vector t is measured with respect to the feature in question. Tracking feature *p* to feature *q* is simply the problem of determining the six parameters that comprise the deformation matrix D and the translation vector t. In the case of pure translation, D will be the identity matrix and thus

$$\delta = p + t \tag{2.37}$$

Because of this, the case of pure translation is computationally simpler and thus preferable due the higher frame rates typically found in video data. Since the motion between adjacent frames of standard video is generally quite small, it turns out that setting the deformation matrix to identity is a safe computation [50], leaving us with the translation vector being exactly the displacement vector. The displacement vector is computed using a pyramid of resolutions because processing a high resolution image is computationally intense. The multi-resolution pyramid within the feature tracker reduces the resolution of the entire image, say by a factor of 2. Tracking occurs by tracking a features general area in the lowest resolution and upgrading the search for the exact location as it progresses up the pyramid to the highest resolution.

The displacement vector *t* is chosen as the offsets that minimize the difference between the windows surrounding the features. The difference is referred to as the residue (ε) and is formally defined by the following double integral over the windows in images I and J:

$$\varepsilon = \iint_W [I(x-t) - J(x)]^2 w dx \tag{2.38}$$

Where w is a weighting function that can be set to one in the simplest scenario, could be Gaussian to allow more weight to the centre of the window or could be more complex to de-emphasize regions of high curvature. When the translation vector *t* is small, the image intensity function can be approximated by a Taylor expansion that is truncated to linear terms:

$$I(x-t) = I(x) - g \cdot t \tag{2.38}$$

and we can re-interpret the residue function (2.38) to be

$$\varepsilon = \iint_W [I(x) - g \cdot t - J(x)]^2 w dx = \iint_W (h - g \cdot t)^2 w dx \tag{2.38}$$

Where h = I(x)-J(x), and g is the image gradients.

This makes the residue quadratic in t and the minimization occurs when the first derivative of the residue is 0. Differentiating the residue and setting it equal to zero results in

$$\iint_W (h - g \cdot t) g w dA = 0 \tag{2.39}$$

and since $(g \cdot t)g = (gg^T)t$ we have

$$\left(\iint_W (gg^T) w dA \right) t = \iint_W h g w dA \tag{2.39}$$

Which is a system of two scalar equations in two unknowns.

For ease of explanation in the next section, equation 2.39 is reconstructed as

$$G t = e \tag{2.40}$$

Where G and e are the from 2.39, and t is the displacement vector.

In plain English: ε is the sum of squared differences of the window gradients, and we are looking to find the matching window (at displacement t) that minimizes ε.

2.11.2 Good Features to Track

The basic strategy of selecting features is to find areas with sufficient texture changes in both the X and Y direction. Often these "interest" features are thought of as texture regions or corners and their detectors *usually* will find good features to track. However, the method looks to find optimal regions to track based on the tracking equation. By removing windows that are not optimal, the features selected are optimal by construction. We are able to track features from frame to frame when equation 2.39 is easily solved.

Effectively this means that the 2x2 coefficient matrix represented by G must be greater than the noise level and well conditioned. Greater than the noise level means that the two eigenvalues of G are large; and well conditioned means that the eigenvalues cannot differ by several orders of magnitude. When the eigenvalues are small, we have a uniform (non-textured) area. A large and small eigenvalue correspond to a unidirectional texture (i.e an edge) and two large eigenvalues represent a bi-directional texture (i.e. either a corner or salt-and-pepper like texture). Only when the two eigenvalues are large, is the equation in 2.39 considered optimal, and therefore can we achieve reliable tracking.

As a result, the window for tracking is acceptable when both eigenvalues surpass a given threshold. When we are selecting a certain number of features (N) to track, we simply take the windows corresponding to the N largest eigenvalue pairs. Furthermore, we can apply some level of non-maximal suppression to ensure that features we are tracking are at least some distance apart.

Chapter 3

Computing Camera Positions from Uncalibrated Video/Image Sequences

3.1 Introduction

Recently, a great deal of research has been done in the field of projective vision [22, 27, 31, 32]. Furthermore, a number of systems have been implemented [33, 34, 35, 36] that can, in theory, compute a 3D model automatically from an uncalibrated image sequence. The idea is to compute photogrammetric information from image sequences without requiring a prior camera calibration process. We believe that there are four primary reasons for the recent rapid advances in the projective framework.

1. Basic theoretical work defining the fundamental matrix, the trilinear tensor and their characteristics.
2. Simple and reliable linear algorithms for computing these quantities from a set of 2D image correspondences.
3. Robust random sampling algorithms for filtering noisy and inaccurate correspondences.
4. Advancement of algorithms for performing autocalibration using only the projective camera positions.

This combination of advances has made it theoretically possible to create a 3D VRML model of a scene from image sequences. Projective methods are normally used to deal with uncalibrated image sequences; however, we believe that even when calibration information is available it is often better to use the projective approach to automatically compute image correspondences and sparse depth information. By computing the fundamental matrix and the trilinear tensor for image pairs and triplets, it is possible to produce a reliable and accurate set of correspondences. When calibration information is available these correspondences can be used directly by a photogrammetric process to compute the camera positions in Euclidean space.

Often it is the case that new research into the field will require the implementation of many of these projective algorithms. These algorithms are often quite complicated to implement and result in many months delay before new research can actually begin. In

this chapter, we describe a system that is freely available for researchers to help facilitate expedient research into the field of projective vision. By this, we hope to make it possible for others to explore and experiment within this paradigm, which we believe will have a significant future influence on the field of computer vision.

Beyond merely describing our experience in re-implementing published core algorithms, this chapter makes a number of other contributions that address some outstanding issues in the field.

1. These programs are not generally available to the public in either source or binary form. As of 2000, the only exception we know of is [36]. The systems described in this work have been made publicly available to facilitate efficient future research and is, to our knowledge, the first publicly available system that computes the trilinear tensor.
2. We show that, in practice, projective methods along with random sampling algorithms solve the correspondence problem for many image sequences and that this is relatively simple, even in the presence of known calibration parameters.
3. We present a way to stabilize the corner selection process, and introduce a simpler relaxation-like methodology based on the idea of disparity gradient.
4. We present a way of dealing with the problem of cumulative error in the tensor computation and demonstrate that projective methods can handle surprisingly large baselines, in certain cases over one third of the image size.
5. We present a way of dealing with non convergence of the bundle adjustment photogrammetric process that is due to minute baselines.

In most papers on projective vision, the goal is to compute a projective reconstruction, assuming that camera calibration information is not available [33, 34]. This is generally followed by an autocalibration process which enables the projective reconstruction to be upgraded to metric (Euclidean) form [17]. The implication is that, if calibration information were available, one should use traditional structure-from motion-algorithms (SFM) to process the image sequence. We claim that this is not necessarily the case. Surprisingly, for most image sequences it is not necessarily easier to compute reliable correspondences when calibration information is available. The reason is that the random sampling algorithms, which are the key to dealing with bad correspondences, are much easier to use in a projective framework than in a calibrated framework [20].

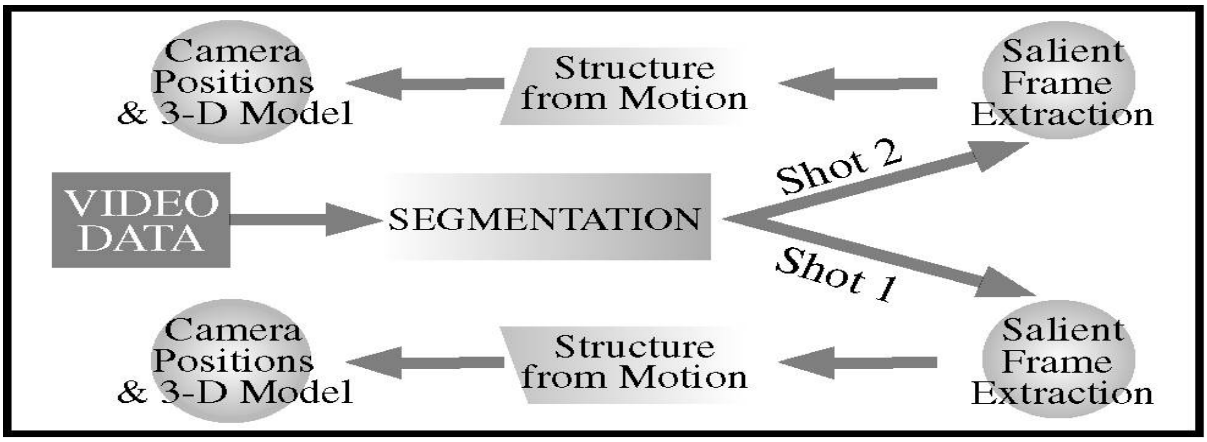


FIGURE 3.1: SYSTEM FOR GOING FROM VIDEO TO 3D CAMERA POSITIONS

Using projective methods in combination with algorithms from the field of robust statistics [37, 38], one can automatically obtain very reliable correspondences for many image sequences, even those with considerable camera motions (i.e. a wide baseline). Producing such accurate correspondences is a multi-step process, where the final result is the trilinear tensor and thus indirectly a projective reconstruction. We take a sequence of images and show that the correspondences that support the trilinear tensors are correct and accurate enough to be input directly to a photogrammetric package to compute a set of 3D camera positions, assuming we have a prior camera calibration, or go through an autocalibration process such as the one presented in Chapter 5. To improve the accuracy of projective reconstruction across an image sequence it is usual to perform a projective bundle adjustment. However, we believe, as do others [39], that the non-linear optimization inherent in the bundle adjustment is better done in metric space than projective space [40]. For this reason, we will not use the tensor for its projective reconstruction, but only to produce a set of accurate image correspondences. In this way, the tensor need only be accurate enough to identify individual matching features in adjacent images; a required accuracy of only a single pixel. Effectively, this means that the cumulative error of the tensor over an image sequence is not an issue.

The correspondences that support the tensor are used as input to a photogrammetric bundle adjustment program to accurately compute camera positions [41]. Once we have these camera positions, it is possible to rectify these images so that the Epipolar lines are horizontal. Then one can compute dense depth maps using traditional stereo al-

gorithms [33]. If the goal is to make 3D models, this is not necessarily the best approach as stereo algorithms will not work in regions without natural texture. For this reason, we believe that it is best to compute dense depth using active methods, since they will succeed even when there is no texture [42]. However, passive methods are sufficient for computing camera positions since this requires sparse, not dense depth, which is much easier to obtain. Furthermore, once these camera positions are known they could be used to rectify 3D data acquired from active sensors that are attached to a rig with the passive cameras. The passive sensors would be used to find the position of the active sensors, which in turn will be used to actually obtain the dense depth necessary to make a 3D model. Therefore our goal is to go from an image sequence to a set of camera positions using only passive technology because much work has gone into rectifying active scanner data [42].

It is important to note that, in practice, this process is divided into two distinct phases.

The first phase computes correspondences from the overlapping image sequence and results in a series of fundamental matrices and trilinear tensors. We dub this stage the correspondence stage, and consists of the following steps:

1. Select images are extracted from image sequences that maximize overlap, with the constraint that baselines cannot be too minute. (§ 3.2.1)*
2. Corner like points are found in each image using a local interest point operator [43, 44] (§ 3.2.2)*.
3. A feature matcher finds a set of potential corner pair matches between two adjacent images in the sequence [36] (§ 3.2.3).
4. These potential matches are pruned using some type of local consistency filter (§ 3.2.4)*.
5. A fundamental matrix is computed from the pruned matches using a random sampling algorithm [14, 20, 34, 45] (§ 3.2.5).
6. Guided matching using the initially computed fundamental matrix (§3.2.6)
7. A set of potential triple matches across three consecutive images are found from the supporting matches from the fundamental matrix (§ 3.2.7).
8. A trilinear tensor is computed from these potential triple matches, again using a random sampling algorithm [46] (§ 3.2.8).

* These marked areas denote that improvements over what is commonly seen in the literature have been made.

Producing the trilinear tensor is equivalent to creating a projective reconstruction of the camera position, along with a projective reconstruction of the matching corner points. However, what is more useful in this work is that the final set of correspondences that support the tensor are in practice, error free, for the vast majority of cases. There are a number of reasons for this result. First, unlike the fundamental matrix, the tensor encodes the constraints among three image pairs. It can therefore produce correct correspondences in the degenerate situation in which the epipolar lines of the two image pairs of the image triple happen to be collinear. The other reason for the reliable results is the use of robust methods to discard bad correspondences. The process begins with a large number of possibly unreliable corner matches and continually prunes these to a smaller set of more reliable matches.

If the final goal is to produce a dense reconstruction of the scene, then once the trilinear tensor is computed the next phase of the process is the reconstruction phase and typically consists of the following steps:

1. Optionally autocalibrate the image sequence (if calibration information is unknown) to allow us move from a projective to a metric reconstruction [17]. Alternative methods of auto calibrating are covered in Chapter 5.
2. Compute camera positions in Euclidean space [41].
3. Rectify the image pairs in the sequence so that the epipolar lines are horizontal and coincident [33].
4. Run a stereo algorithm to compute dense depth from the rectified image pairs [33].

The set of steps in this last phase makes a number of assumptions, and have been well addressed over the years as calibrated stereo vision problems. The first assumption is that the goal is actually to create a dense 3D metric reconstruction of what has been viewed by the image sequence. However, for some applications, the output of the first phase, a set of projective camera positions, may be sufficient. An example occurs the field of augmented reality in which the goal is to place synthetic objects in an image of a real scene. In this case, the computed tensors can be used to place these synthetic objects in appropriate positions without having either dense depth or metric camera positions. As previously noted, we believe that for obtaining dense depth it is best to use active, not

passive methods. However, for obtaining the position of an active sensor it is feasible to use only an image sequence from a passive co-mounted sensor [47]. Also, in many model building applications it is not difficult to obtain camera calibration, and we assume this information is available either directly or from an autocalibration routine such as the one presented in Chapter 5.

Our goal is to find the 3D camera positions from an image sequence using projective methods to solve the correspondence problem and well known methods [41] to compute the 3D reconstruction. The details of the procedure are described in the next section.

3.2 Processing Steps

We now describe the details of the process that takes a video image sequence and computes a set of 3D camera positions. In doing so, we highlight the changes and additions that have been made over what is described in the literature.

3.2.1 Selecting Frames that are well suited

We have noticed that the photogrammetric bundle adjustment software [41] will sometimes not converge if the spacing between the image pairs is too minute. This is often the case with video data. A good selection of frames from a video sequence can produce a much better set of input image data so that the bundle adjustment algorithms are more likely to converge and therefore ensure a more reliable reconstruction. Due to the wide availability and simplicity of use, video cameras are ideal image acquisition devices. The high frame rate ensures that full coverage of the scene is possible; however this advantage is surprisingly a disadvantage as well. The large volume of frame data is not only impractical to process in a timely manner, but the minute baselines between frames can also cause problems during the bundle adjustment phase of the structure from motion (SfM) algorithms. The method described here is a novel approach to preprocessing video image sequences to select a sub-sequence from larger sequence of video frame data. Based on a proven tracking mechanism, the algorithm remains quite simple yet effective for identifying and extracting salient frame data for subsequent use in computing camera positions.

The obvious approach of regular frame sampling (effectively reducing the frame rate) shows its inadequacies quickly. Due to banding and interlaced video, the regularly selected frames may not be ideal for image processing. Another problem is that frames selected in this manner have not been picked for their suitability to the structure from motion problem, but rather on a frame rate assumed to be good. The structure from motion algorithms work best on images with large overlap to allow for good feature matching yet significantly large baseline to ensure parallax large to enough to keep the problem well conditioned. By simply changing the frame rate it is clear that the images produced by this method may cause the structure from motion algorithms to be ill conditioned. High frame rates increase the chance that parallax will not be sufficient and low frame rates reduce the amount of overlap required to adequately match features. Clearly selecting a fixed frame rate is not an effective approach to salient frame extraction for the structure from motion problem. In fact, the ideal frame rate turns out to be variable, depending on the two factors that make the structure from motion problem well conditioned: overlap and parallax.

3.2.1.1 Motion Estimation and Feature Tracking

The feature tracker we use is based on the early work of Lucas and Kanade [48] that was developed fully by Tomasi and Kanade [49], Shi and Tomasi provide a complete description [50] that is readily available. Recently, Tomasi proposed a slight modification, which makes the computation symmetric with respect to the two images; the resulting equation is fully derived in [51]. Briefly, features are located by examining the minimum eigenvalue of a 2x2 image gradient matrix that is noticeably very similar to the Harris corner detector [44]. The features are tracked using a Newton-Raphson method of minimizing the difference between the two windows.

We continue by presenting a very brief outline of the work by Tomasi etal [48,49,50,51]. Given a point p in an image I, and its corresponding point q in an image J, the displacement vector δ between p and q is best described using an affine motion field:

$$\delta = Dp + t \tag{3.1}$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \tag{3.2}$$

is a deformation matrix is a Hessian matrix, and t is the translation vector of center point of the tracked feature window. The translation vector t is measured with respect to the feature in question. Tracking feature p to feature q is simply determining the six parameters that comprise the deformation matrix D and the translation vector t.

Clearly in the case of pure translation, D will be the identity and thus

$$\delta = p + t \tag{3.3}$$

Because of this, the case of pure translation is computationally simpler and thus preferable. Since the motion between adjacent frames of standard video is generally quite small, it turns out that setting the deformation matrix to identity is the safest computation [50], leaving us with the translation vector being exactly the displacement vector. A complete explanation of the tracking equations is given in Chapter 2, Section 2.11

In the preprocessing system described in Section 3.2.1, our goal is to monitor the parallax and overlap between frames in order to ensure the stability and well conditioning of the structure from motion algorithms. Monitoring the motion through lost features and feature parallax via feature tracking allows us to decide when there is suitable parallax and overlap between frames for the structure from motion algorithms. The exact criterion for extracting two frames to be fed into the structure from motion algorithms is described in the next section.

3.2.1.2 Salient Frame Extraction

Once the input video sequence has been segmented into its individual shots, and a complete description of once such method is presented in Chapter 4, each shot can then be independently processed to extract salient frames and then further processed using the correspondence and reconstruction phases of the system. Since the salient frame extraction the structure from motion parts of the system are independent, this processing is distributable and easily made parallel.

Briefly, the extraction is done by selecting a set of features smaller than the set used by the structure from motion algorithms and tracking them across adjacent frames. A salient frame is signaled when enough features have surpassed a certain user specified parallax and/or enough features have disappeared and can no longer be tracked. This criterion is exactly what is required to ensure the success of the structure from motion algorithms.

In algorithmic form, salient frame extraction

```
1. Select good features for frame 1 place in feature list FL
2. For each frame x in the video
   a. Track features from FL in current frame x
   b. Count number of lost features
   c. Count number of features that have passed the parallax threshold
   d. If detecting boundaries (cuts)
        i. If (features lost > boundary threshold)
             1. Signal boundary and extract boundary frame
             2. Refresh the feature list FL using boundary frame
       ii. Endif (i)
   e. Endif (d)
   f. If (features lost + features over parallax threshold) > threshold
        i. Signal & extract frame
       ii. Refresh feature list FL using current frame x
   g. Endif (f)
3. Endfor (2)
```

The number of features to track the parallax threshold will vary depending on video dimensions, however a good rule of thumb is the following: 25 features for every 10000 pixels, the parallax threshold should be 1/8 of the smallest dimension, and a sensitivity threshold of 75 percent. For example, a video that is 320x240 would have 192 features to track, a parallax threshold of 30 pixels (240/8), and a sensitivity threshold of 75 percent. This will supply images with significant overlap and sufficient parallax. When detecting images, a boundary threshold of 95% or greater is sufficient.

3.2.1.3 Salient Frame Extraction Results

A series of small video clips was created to test the accuracy and capabilities of the algorithm. These master clips consist of three smaller subsequences that are cut together. These sequences were created using a standard analog video camera commonly found in many stores and digitized using a video capture card and converted to an MPEG sequence.

Master Sequence	Sub-Sequence	Frame count	Selected frames	Reduction rounded to nearest %
1	Medical Centre	350	13	96
1	Warehouse	336	13	96
1	Body Shop	153	6	96
2	KFC	207	6	97
2	Caisse Populaire	252	10	96
2	Doctors Office	319	5	98
3	House 1	333	16	95
3	House 2	542	15	97
3	House 3	369	12	97
4	Play structure	653	25	96
4	Little House	662	20	97
4	Slide	375	12	97
5	Barn **	374	26	93
5	Temple 1	481	7	99
5	Temple 2 **	429	36	92

Table 3.1: Frame reduction for image sequences

The cut detection capabilities proved flawless and correctly identified all sequence start and end points. Surprisingly, this was more accurate than a pixel level cut detection mechanism used to initially verify the sequence start and end points. The pixel level cut detection missed one of the sequence starting points. We explore these capabilities in detail in Chapter 4.

** the sequence was taken from a moving vehicle, the higher camera speed results in more frames being selected as an overall percentage.

FIGURE 3.2: EVERY 2ND EXTRACTED FRAME FROM EXAMPLE SEQUENCE

As one can see from Figures 3.2 and 3.3, the spacing of the extracted frames is very consistent and regular. These baselines also make the structure from motion algorithms well conditioned and hence the images taken from the sequences are well suited by construction..

FIGURE 3.3: ALL FRAMES FROM EXAMPLE SEQUENCE

3.2.2 Finding corners/interest points

The next step is to find a set of corners or interest points in each extracted image. These are the points where there is a significant change in image gradient in both the x and y direction. We offer the use of Smallest Univalue Segment Assimilating Nucleus (SUSAN) corner detector [43] or the more commonly used Harris corner detector [44]. Studies have shown the Harris detector to be the more stable of the two [52].

Both methods typically require a user selected threshold to determine whether or not a pixel and its surrounding area represents a corner. Instead of setting a corner threshold, we return a fixed number of corners from each method by sorting the appropriate gradient values, that have had a non-maximal suppression operator applied, and return the top N strongest corners. This relatively simple addition to the standard corner detectors tend to stabilize the results when the images have differening contrast and brightness because the proper threshold is selected automatically and only the strongest corners are returned. Furthermore, the running time of the sorting algorithm represents the upper bound on this addition to the corner finding. The final results are not particularly sensi-

tive to the number of corners that the user finds, and typically we use in the order of 800 corners selected from an image. This stage returns a set of corners C_i for image i in the sequence.

3.2.3 Matching Corners

The next step is to match corners between adjacent images. A local window around each corner is correlated (using normalized cross correlation at a sub-pixel level) against all other corner windows in the adjacent image that are within a certain distance that represents the upper bound on the image disparities. We generally set this upper bound to approximately 1/3 the length of the longest dimension. Any corner pair within the specified image disparity that passes a minimum correlation threshold is flagged as a potential match. This gives us a one-to-many relationship in the putative match set which is complicated. We reduce the complexity by enforcing a simple symmetry test.

The symmetry test is effectively reducing the one-to-many relationship to a one-to-one relationship where corners strongly match one another. For example, consider a corner p in the left image, and a corner q in the right image of an image pair. Assume that the strongest match for p in the opposite image, the right image, is labeled Right(p). Similarly for q the strongest match in the left image, is labeled Left(q). The symmetry test requires the correlation be maximal in both directions. In other words a match (p; q) is acceptable if and only if p = Left(q), and q = Right(p).

The symmetry test reduces the number of possible matches significantly and forces the remaining matches to be one-to-one. The total number of possible matches between images is therefore less than or equal to the total number of corner points. Without the symmetry test constraint there are far more matches; but these matches are much less reliable. For wider baseline images, it is useful to relax the symmetry test and to accept the n best matches (usually in the order of 4). In this case we still require that the results be symmetric, that is that each of these matches actually be one of the n best in a symmetric fashion.

3.2.4 Local consistency filtering

The next step is to perform some type of local filter on these matches. The idea is that just by looking at the local consistency of a match relative to its neighbours it is possible to prune many obvious false matches. This is not always done in the literature, but is a sensible step, since the computational cost of using a local filter is low. One possible approach to prune matches is to use relaxation [14]. We use a simpler relaxation-like process to prune false matches, one based on the concept of disparity gradient [53]. The disparity gradient is a measure of the compatibility of two correspondences between an image pair. Assume the first correspondence maps a corner point p_l ($x1_{left}$; $y1_{left}$) in the left image to another corner point p_r ($x1_{right}$; $y1_{right}$), in the right image. Similarly, a second correspondence maps corners ($x2_{left}$; $y2_{left}$) into ($x2_{right}$; $y2_{right}$). The disparity of these two correspondences are the vectors d_1 = ($x1_{left} - x1_{right}$; $y1_{left} - y1_{right}$) and d_2= ($x2_{left} - x2_{right}$; $y2_{left} - y2_{right}$). The cyclopean separation of the vectors d_1 and d_2 ($cs(d_1,d_2)$)is defined as the vector that joins the midpoints of d_1 and d_2. The disparity gradient is the ratio of the magnitude of the difference of the two disparity vectors d_1 and d_2 and the magnitude of the cyclopean separation

$$disp.grad = \frac{|d_1 - d_2|}{|cs(d_1, d_2)|} \tag{3.4}$$

Corner points that are close together in the left image should have similar disparities, and the disparity gradient is a measure of this similarity. Thus, the smaller the disparity gradient, the more the two correspondences are in agreement and vice-versa. The disparity gradient measure has been used in some calibrated stereo algorithms to prune invalid correspondences. Typically, these algorithms reject any correspondence with a disparity gradient greater than 1.5. In our case, we compute the disparity gradient of each correspondence with respect to every other correspondence. The sum of all these disparity gradients is a measure of how much this particular correspondence agrees with its neighbours.

We iteratively remove correspondences until they all satisfy the condition that the correspondence with maximum disparity gradient sum is within a small factor (usually 2.0) of the correspondence with minimum disparity gradient sum. Using this simple disparity gradient heuristic we are able to remove significant numbers of bad correspon-

dences at a very low computational cost. Typically, at least 40% of the total number of incorrect correlation matches is removed by this process. There is an additional benefit derived by performing this localized filtering: Efficiency in the next step. By reducing the number of false matches in the set the random sampling process requires fewer samples to converge to correct answer. This is because the numbers of correct matches appear in a higher proportion and random selection of a subset of these matches will result in a higher proportion of the selected matches being correct. The number of iterations (n) required to get a good fundamental matrix is given by the following equation:

$$n = \frac{\log(1 - \Pr(Fgood))}{\log(1 - p^8)} \tag{3.5}$$

where Pr(Fgood) is the probability of computing a good fundamental matrix F, and p is the proportion of good matches in the putative match set. This step results in a set of putative matches DM_{ij} for image pair i and j.

3.2.5 Computing the fundamental matrix

The original matches between image i and j produced by the correlation process are labeled as the set M_{ij} , and the filtered matches that pass the disparity gradient test as the set DM_{ij}. The next step is to use these filtered matches to compute the fundamental matrix which is the uncalibrated version of the essential matrix. This process must be robust, since it can not be assumed that all of the correspondences in DM_{ij} are correct. Robustness is achieved by using concepts from the field of robust statistics, in particular, random sampling [10, 11, 12, 21, 23] as outlined in Chapter 2. A fundamental matrix, F_{ij}, is then computed from this minimal set. The set of all corners that support this fundamental matrix is called the support set SF_{ij}. The fundamental matrix F_{ij}, with the largest support set SF_{ij} is returned by the random sampling process.

While this fundamental matrix has a high probability of being correct, it is not necessarily the case that every correspondence that supports the matrix is valid. This is because the fundamental matrix encodes only the epipolar geometry between two images. A pair of corners may support the correct epipolar geometry by accident (this is known as degeneracy []). This can occur, for example, with a checkerboard pattern when the epipolar lines

are aligned with the checkerboard squares. In this case, the correctly matching corners can not be found using only epipolar lines (i.e. computing only the fundamental matrix). This specific type of ambiguity can be dealt with by computing the trilinear tensor.

The PVT supports the computation of the fundamental matrix from a variety of different algorithms, including Affine, Hartley's 8 point algorithm, Phil Torr's algorithm, and Kanatani's renormalization procedure. Furthermore the toolkit allows the computation of homographies for planar warps and rotations.

3.2.6 Guided matching

Once a putative fundamental matrix has been computed we revert back to matching corners phase as outlined in section 3.2.3. Again we use normalized cross correlation but we restrict our matching criteria even further by only looking for putative matches that fall near the epipolar lines defined by the previously computed fundamental matrix. This back step allows us to generate a new set of correspondences have a higher probability of being a proper corresponding pair. One the new putative guided match set GM_{ij} has be computed, we again perform a disparity gradient filter followed by a final computation of the fundamental matrix (F_{ij}) that in practice gives us a larger support set (SF_{ij}) than was computed by the previous stage.

3.2.7 Computing putative triple correspondences

This stage is a simple matching using the transitive property of equality. In practice, 10 to 25 percent of these putative triplets are not exact matches and are often mismatched by several pixels. These putative triple correspondences are then used to compute the trilinear tensor in a random sampling process. We compute the trilinear tensor from the correspondences that form the support set of two adjacent fundamental matrices in the image sequence. Consider three adjacent images, i, j and k and their associated fundamental matrices F_{ij} and F_{jk}. Each of these matrices has a set of supporting correspondences, which we call SF_{ij} and SF_{jk}. Say a particular element of SF_{ij} is (x_i y_i ; x_j y_j) and similarly an element of SF_{jk} is (x'_j y'_j ; x_k y_k). Now if these two supporting correspondences overlap, that is if (x_j y_j) equals (x'_j y'_j) then the triple created by transitivity then is

a member of PT_{ijk}, the putative triplet set. The set of all such possible supporting triples is the input to the random sampling process that computes the tensor.

3.2.8 Computing the trifocal tensor

The trilinear tensor relates the image coordinates of matching corners in three images instead of two images. It is therefore inherently a more stable, and a more discriminating quantity that the fundamental matrix [31]. We use the putative triple matches, PT_{ijk}, over three views to robustly computer the trifocal tensor using techniques outlined in Chapter 2. The result is the trifocal tensor T_{ijk}, for three views i,j,k, computed using a random sampling method [21]. Furthermore a set of triples (corner in the three images) that actually support the computed tensor is output, which we call ST_{ijk}. The toolkit allows the computation of an affine tensor, and projective tensors using methods outlined by Hartley [18] or by Torr [23].

3.2.9 Computing the 3D information

We have gone from a set of corner points C_i; C_j , and C_k; to a set of matches M_{ij} , and M_{jk}; to a set of filtered matches DM_{ij} , and DM_{jk}; to a pair of fundamental matrices F_{ij}; F_{jk} and support SF_{ij} and SF_{jk}; to a set of putative triplets PT_{ijk}, to a tensor T_{ijk} with support ST_{ijk}. The cardinality of each of the supporting match sets always decreases, but the confidence that each match is correct increases. The entire process begins with many putative matches, and refines these to a few high confidence matches. The final matches ST_{ijk} that support the tensor T_{ijk} range in cardinality from 20 to 100, and in practice, have a very high probability of being correct. As we stated in the introduction, the goal is to compute the 3D camera positions from the image sequence, not to compute dense depth. We therefore do not perform the steps in the second phase; rectification and stereo. Instead, we take the correspondences that support the overlapping tensors and send them to a photogrammetric bundle adjustment program [41]. Assume that we have a sequence of images numbered from 1 to n, and have computed a set tensors T_{123}, T_{234}, … $T_{(n-2)(n-1)(n)}$. Consider the tensors T_{ijk} and T_{jkl} which have supporting correspondences (x_i y_i , x_j y_j , x_k y_k) in ST_{ijk} and (x'_j y'_j , x'_k y'_k , x'_l y'_l) in ST_{jkl}. Those correspondences for which (x_j ; y_j ; x_k; y_k) equals (x'_j ; y'_j ; x'_k ; y'_k) represent the same corner in images *i, j, k* and *l*. This corresponding corner list is then sent directly to the commercial bundle adjustment program

Photomodeler [41] which, as a commercial product, uses established algorithms. Since we know the camera calibration, either aproiori or via an autocalibration method such as the one as outlined in Chapter 5, we use these correspondences, the calibration information and Photomodeler to compute the 3D camera positions, along with the 3D coordinates of the matching features.

3.3 Experiments

Over 20 experiments have been conducted under a variety of lighting conditions and camera motions. Some of these experimental data samples come from the computer vision literature, while others have created using modern digital cameras. All of the experimental results can be found as a part of the Projective Vision Toolkit example sets[2]. Due to limitations in space, only a few are presented in this thesis, particularly those examples that are found in the previous literature because the capture process was not under our control.

In our first example we begin with a complete video sequence taken of an indoor scene. The camera operator is performing some "sky writing" motions. The video sequence, which has 1400 frames, has the frame extraction process described in section 3.2.1 executed to reduce the frames in the sequence down to only 81. In figure 3.4 we see a selection of 9 frames from the reduced original video sequence. We proceed to run the 81 selected frames through the modular process described above to automatically solve the correspondence problem for the selected subsequence. The correspondences and the camera calibration parameters are fed into the structure from motion software and the 3D reconstruction of the 2355 correspondence points and the 80 camera positions are computed. As we can see in Figure 3.5, the "sky writing" experiment is effectively captured when the camera positions and points are reconstructed in a virtual rendering of the 3D points and cameras and spell out the letters N, R, and C.

[2] The Projective Vision Toolkit webpages can be found here: http://cg.scs.carleton.ca/~awhitehe/PVT/

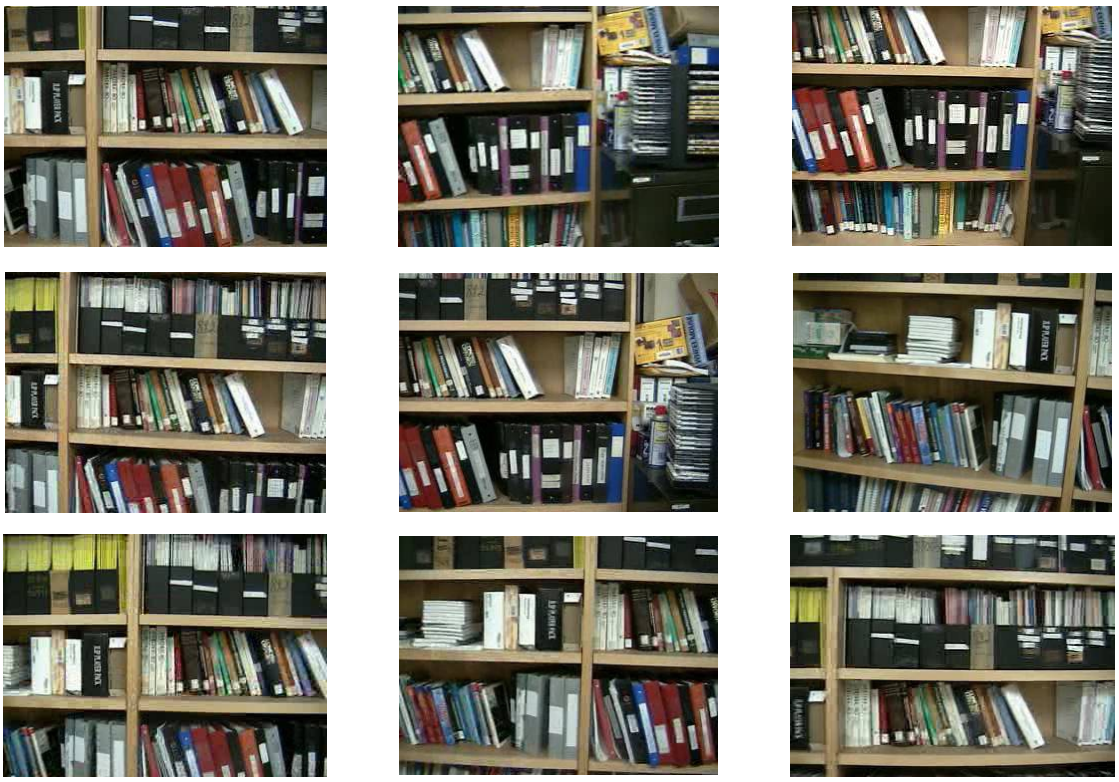


FIGURE 3.4 SELECTION 9 OF IMAGES FROM VIDEO SEQUENCE

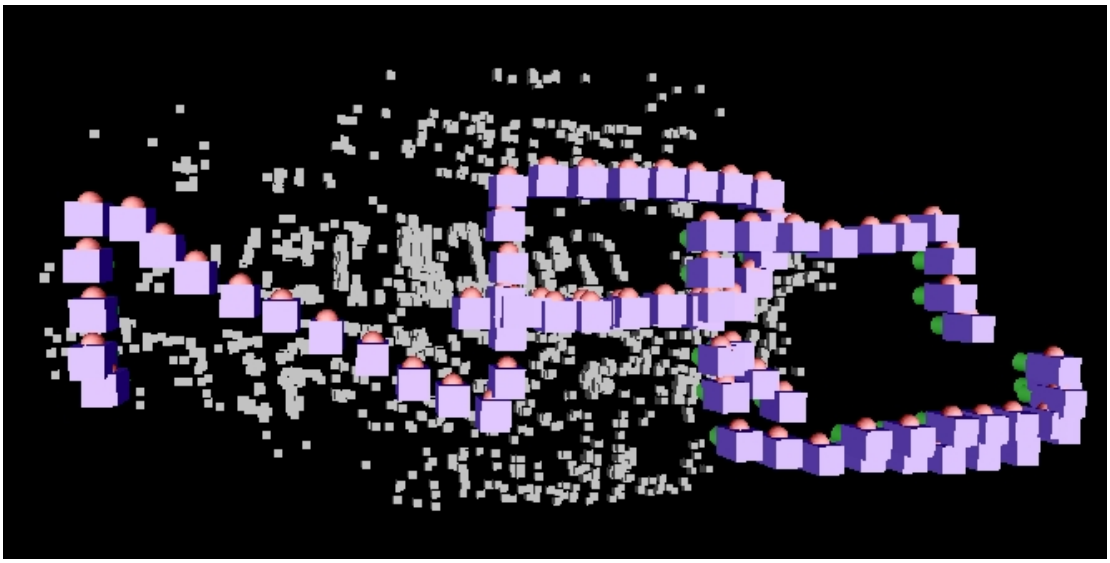


FIGURE 3.5 RECONSTRUCTED CAMERA POSITIONS AND POINTS FROM A VIDEO CAMERA

Attempts to run the entire 1,400 frames through the process and then into the structure from motion algorithms would be computationally intense and the bundle adjustment phase would have problems converging due to the minute baselines present that are caused by the high video frame rate.

In our next example, we present results using the well-known Carnegie Mellon House sequences. In this sequence, the camera was moved with deliberate fixed motions that offer us a known ground truth. In the tables below, we see how the number of matches decreases as we step through each of the previously described phases of computation.

FIGURE 3.6 3 SAMPLES FROM THE CMU-BIGHOUSE SEQUENCE, FRAMES 1,6 & 11.

For two views, we have an approximate 10 percent drop in matches for each stage, and a 50 percent drop when we generate putative triples and finally another 20 percent drop for those triples that support the computed tensor.

Image Pair	Correlation Matches	Locally filtered Matches	Fundamental Matches
1-2	417	389	363
2-3	349	321	313
3-4	470	447	420
4-5	487	479	455
5-6	393	373	366
6-7	531	530	518
7-8	422	405	393
8-9	509	506	494
9-10	505	504	487
10-11	433	411	398

Table 3.2 Match counts for pair wise phases

Image Triplet	Putative Matches	Tensor Support Matches
1-2-3	185	156
2-3-4	209	196
3-4-5	279	236
4-5-6	269	236
5-6-7	278	168
6-7-8	311	243
7-8-9	310	238
8-9-10	378	353
9-10-11	287	225

Table 3.3 Match counts for triplet phases

Once we have computed these high confidence correspondences, we proceed by passing the correspondence information and camera calibration information (obtained either via a calibration process or autocalibration) into the photogrammetry software to compute the camera positions and sparse depth information for the corresponding points. As we can see in Figure 3.7, the major structures of the scene such as the peaks in the roof are quite evident. Furthermore, the camera positions show the deliberate well conceived motions.

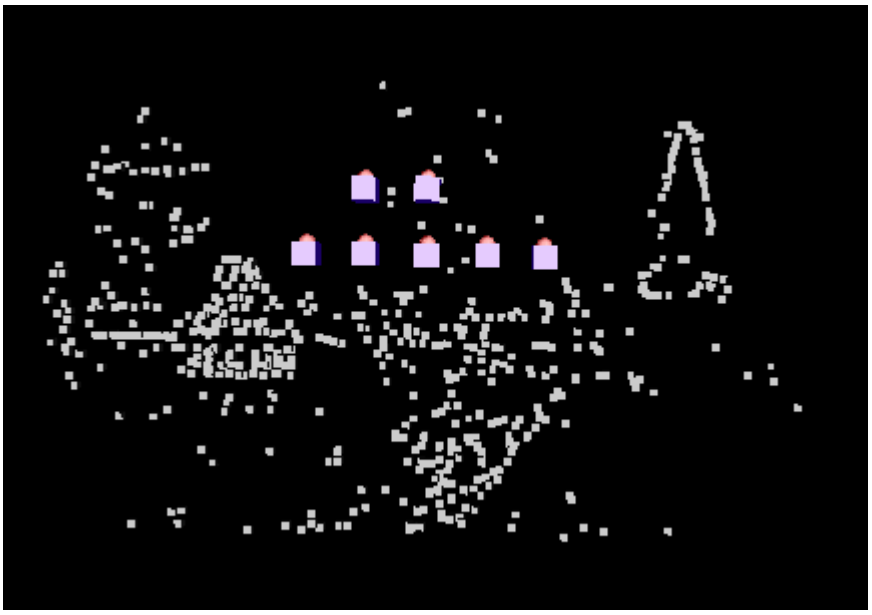


FIGURE 3.7 RECONSTRUCTED CAMERA POSITIONS AND FEATURE POINTS

For those not familiar with the CMU-Bighouse sequence, the cameras were deliberately moved forward and backward in certain cases and the cameras are not visible in the reconstruction as shown in Figure 3.7. However as we can see in Figure 3.8, a close up view of the cameras shows the backs of the cameras in their original positions. However the rendering of the cameras in their new positions effectively overwrites the front of the cameras in the original position.

Finally, in Figure 3.9 we present an image taken from the sequence with the reconstructed points in black re-projected and overlaid onto the image. The camera posi-

tions are also re-projected back onto the image plane, and we can see that the points are quite accurately projected to their original locations.

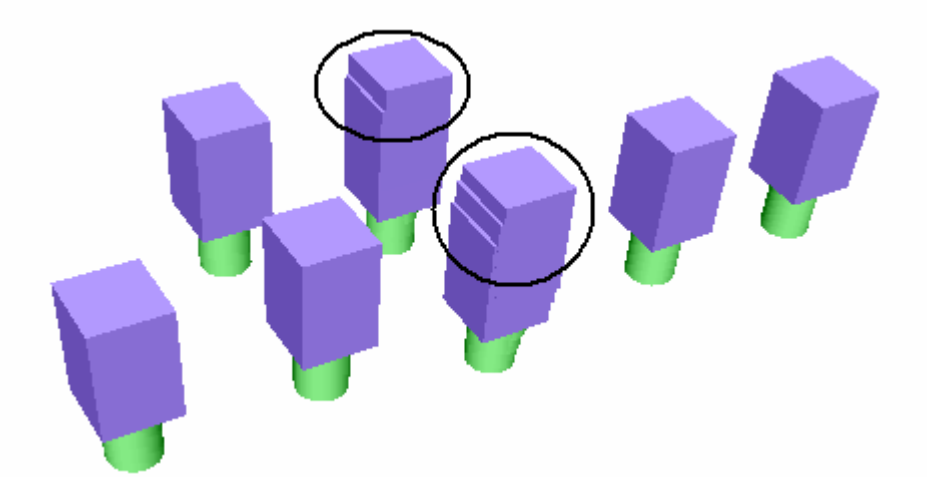


FIGURE 3.8 RECONSTRUCTED CAMERA POSITIONS WITH OVERLAPS HIGHLIGHTED

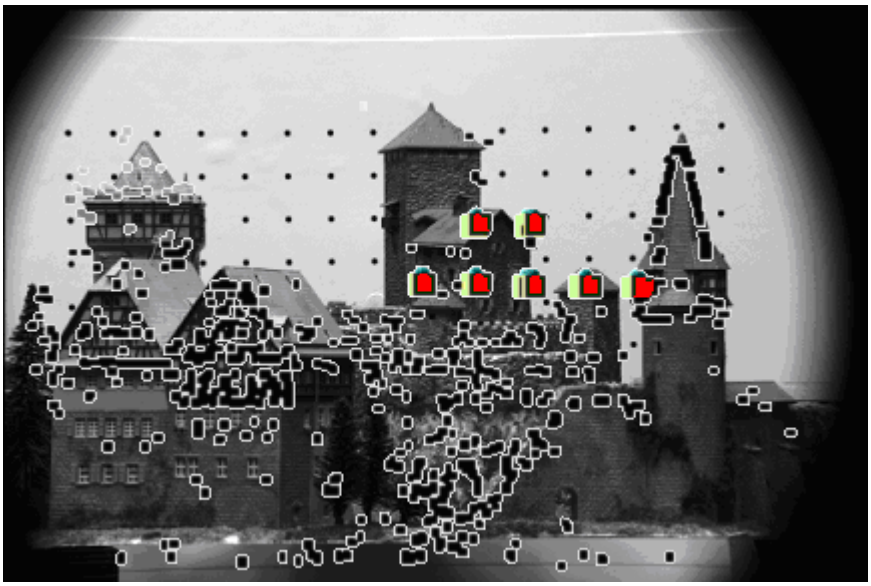


FIGURE 3.9 ORIGINAL IMAGE WITH FEATURES AND CAMERAS OVERLAID

In our final example, we have another well known sequence, the Oxford basement. This example is particularly difficult because the camera is moving in a forward motion that similar to that of a change in focal length along the Z-axis. An interesting effect to note here is that because the focus of expansion falls within the image plane, the localized filtering does not appear to be useful way to prune false matches. Upon reflection of disparity gradient, this result makes sense because the magnitude of the individual corre-

spondence vectors decrease in size as they near the location of the focus of expansion. This will result in the cyclopean separation generally being much larger than the magnitude of the difference of the two vectors. This will result in very small disparity gradient values and therefore prevent pruning of correspondences.

FIGURE 3.10 EXAMPLE IMAGE WITH OVERLAID CAMERA POSITIONS AND POINTS

Image Pair	Correlation Matches	Locally filtered Matches	Fundamental Matches
0-1	281	280	241
1-2	288	286	263
2-3	291	289	236
3-4	273	272	233
4-5	277	276	240
5-6	254	252	219
6-7	279	275	242
7-8	238	232	145
8-9	229	226	148
9-10	256	252	212

Table 3.4 Match counts for pair wise phases

Image Triplet	Putative Matches	Tensor Support Matches
0-1-2	198	167
1-2-3	184	134
2-3-4	171	140
3-4-5	180	135
4-5-6	170	144
5-6-7	168	132
6-7-8	114	66
7-8-9	79	46
8-9-10	103	54

Table 3.5 Match counts for triplet phases

As we can see from Table 3.4 the disparity gradient filtering results in very little change in the number of correspondence, while the support sets for both the fundamental matrix and the tensor reduce the correspondence count by approximately 10-20% each.

Finally, we conclude the experiments section with a tabular review of all the experiments conducted and presented as part of the Projective vision toolkit. The exact match counts for each image pair and triplet can be found in Appendix A.

Sequence	Average Correlation Matches	Average Filtered Matches	Average Fund. Support Matches	Average Putative Triple Matches	Average Tensor Support Matches
Ex1a	452	437	421	278	228
Ex1b	293	214	192	57	44
Ex1c	239	123	81	22	18
Ex1d	236	204	148	44	26
Ex2a	185	166	141	31	25
Ex2b	140	138	86	12	12
Ex2c	276	230	115	32	23
Ex3a	233	171	94	24	20
Ex3b	215	173	124	44	35
Ex3c	375	325	157	54	29
Ex3d	343	264	222	104	71
Ex4a	270	218	148	48	36
Ex4b	222	181	90	18	14
Ex4c	175	166	45	7	6
Ex5a	267	264	218	152	113
Ex5b	322	238	199	108	73
Ex5c	193	153	135	61	45

Table 3.6 Reduction in feature correspondence for PVT example sets.

3.4 Conclusions and Discussions

We have presented a modular system for computing a reconstruction of the camera positions from an image sequence. Since our goal is to find the metric camera positions we do not need to create a dense 3D reconstruction. We assume that we have camera calibration information available, but we do not use this calibration information when computing the correspondences. Instead reliable correspondences are computed using the uncalibrated projective method. However, the calibration information is used for computing the 3D camera positions from these correspondences. The final correspondences, those that support the trifocal tensor, are error free in the vast majority of cases. The re-

sults are demonstrated experimentally on a number of examples. There is no doubt that reliable results can be obtained for a wide variety of images, both indoor and outdoor.

In performing our experiments we have drawn some conclusions about the best approaches for each step of the projective reconstruction process. We have described a novel method for extracting a manageable subset of frames from a video sequence and we have also described a way to locally filter invalid correspondences based on the disparity gradient. We believe that projective methods in combination with random sampling solve the correspondence problem for many image sequences. The support set of the fundamental matrix and trifocal tensor are correct correspondences in the vast majority of cases. If the goal is to compute the metric camera positions and the camera calibration is known, we believe that it is best to send the supporting correspondences of the tensor directly to photogrammetry software. Our justification is that a bundle adjustment process is necessary to compute the camera positions accurately and we believe that this is better done in metric space, rather than projective space.

The software used in these experiments has been made publicly available to facilitate expedient future research into the field. To our knowledge, this is the first publicly available software that allows the computation of the trifocal tensor. The toolkit runs on most Unix systems, along with Windows NT/2000/XP/98. The input is a sequence of overlapping images, and the output is a series of fundamental matrices and trifocal tensors, pair wise and triplet correspondences. A more complete description of the toolkit is given in Appendix C.

Chapter 4

Segmenting Video Sequences

4.1 Introduction

Recently, investigation into shot boundary detection schemes has gathered much momentum [54-63]. Cut detection is seemingly easily solved by an elementary statistical examination of inter-frame characteristics; however a truly accurate and generalized cut detection algorithm still eludes researchers. Reliable shot boundary detection forms the cornerstone for video segmentation applications as shots are considered to be the elementary building blocks that form complete video sequences. Applications such as video abstraction, video retrieval and higher contextual segmentation all presuppose an accurate solution to the shot boundary detection problem [60, 64, 65, 66]. Automatic recovery of these shot boundaries is an imperative primary step, and accuracy is essential.

Shot transitions can be classified into four classes based on the 2D image transformations applied during transition production [54].

- *Identity class*: Neither of the two shots involved are modified, and no additional edit frames are added. Only hard cuts qualify for this class.
- *Spatial Class*: Some spatial transformations are applied to the two shots involved. Examples are wipe, page turn, slide, and iris effects.
- *Chromatic Class*: Some color space transformations are applied to the two shots involved. Examples are fades and dissolve effects.
- *Spatio-Chromatic Class*: Some spatial as well as some color space transformations are applied to the two shots involved. All morphing effects fall into this category.

In this work, we concentrate only on the identity class, as our goal is to improve the accuracy of cut detection by introducing a new differencing metric based on stable feature tracking from frame to frame. The basic idea behind the technique has been shown to detect fades [62], but we concentrate solely on cuts in this work.

4.2 Additional Background

In 1965, Seyler developed a frame difference encoding technique for television signals [67]. The technique is based on the fact that only a few elements of any picture change in amplitude in consecutive frames. Since then much research has been devoted to video segmentation techniques based on the ideas of Seyler. Much work has been completed in the area of scene detection, shot detection and annotation and as a result, the methods and algorithms are quite mature. However, a truly accurate cut detection algorithm has yet to be introduced. Any improvements in cut detection will ultimately improve the applications that rely on it.

Hard cuts are the most common transition between shots. A hard cut is the direct concatenation of two shots without the presence of transitional frames. Formally, the resulting sequence $S(x,y,t)$is composed by joining two shots $S_1(x,y,t)$ and $S_2(x,y,t)$ and is characterized by the following:

$$S(x, y, t) = [[1 - u_{-1}(t - t_{hardcut})] \cdot S_1(x, y, t)] + [[u_{-1}(t - t_{hardcut})] \cdot S_2(x, y, t)] \qquad (4.1)$$

where $t_{hardcut}$ denotes the time stamp of the first frame after the hard cut and $u_{-1}(t)$ is the unit step function *(1* for $t \geq 0$, 0 otherwise). [56]

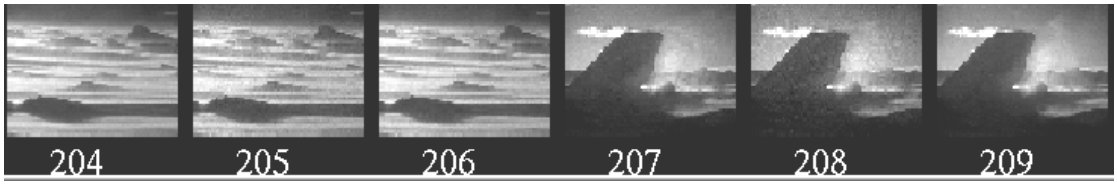


FIGURE 4.1: HARD CUT BETWEEN FRAMES 206 AND 207 OF A VIDEO SEQUENCE.

A hard cut produces a visual discontinuity in the video stream as we see in Figure 1. Existing hard cut detection algorithms differ in the feature(s) used to measure the inter-frame differences and in the classification technique used to determine whether or not a discontinuity has occurred. However, they almost all define hard cuts as isolated peaks in the features time series. In [56] a complete survey is given on techniques to compute inter-frame difference and classify the type of transition. A variety of metrics have been suggested to work on either raw video or compressed data and we briefly outline methods

that have been used in the past, or are currently in use, forming the basis of our comparisons. We will briefly outline the techniques next.

4.2.1 Quantifying Inter-frame Differences

The basic idea behind shot/scene detection is to evaluate the similarity of adjacent frames using some metric. When the similarity measures cross a certain threshold, a scene change or shot boundary will be classified as occurring. By selecting a better method to quantify the inter-frame differences results in the classification algorithm becoming more accurate and easier to implement. In this section we outline several known inter-frame difference quantification techniques.

4.2.1.1 Individual Pixel Differences

Equations (2) and (3) describe a pixel level change metric and a cut classifier respectively.

$$DI_i(x,y) = 1 \text{ if } |I_i(x,y)-I_{i+1}(x,y)| > t$$
$$0 \text{ otherwise} \tag{4.2}$$

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}\frac{DI_i(x,y)}{(X*Y)} > T \tag{4.3}$$

In (4.2), we compute the difference between pixel values between images i and i+1 to create a difference image DI, where t is a threshold signifying individual pixel difference. We then compute the overall image difference using (4.3). If the percentage of image change is greater than a threshold *T*, we declare a shot boundary. The pixel level detection metric displayed in (4.2) and (4.3) is the most basic form for raw, uncompressed video.

Unfortunately, this simple metric measure is very susceptible to object and camera motion. Even if camera motion is compensated and pixels are transformed before being compared, object motion still poses significant difficulties. More sophisticated methods use optical flow, the number and distribution of motion vectors and the strength of the residual derived by block matching as features [57, 58]. In addition, performance of the segmentations relies directly on the adequate selection of two threshold values.

4.2.1.2 Intensity/color histograms

Histogram change metrics utilize histogrammed values of the pixel data rather than the pixel values themselves. This makes the entire system more robust to noise and small object motion. There exist many different histogram possibilities that could be used as there exist many different color spaces. Color spaces such as RGB, YUV, HSV, HIS, YIQ, Lab, Luv, Munsell and opponency colors can all be easily converted from one to another. As such they can be considered equivalent. In practice, simple histogram differencing has shown to be capable and quite efficient. Performance capabilities have been outlined in [56, 57]. Specific examples of histogram techniques are presented in [67, 68, 69]. Note that in general, for histogram techniques, greater improvements in cut detection performance can be attained by making a proper choice for the categorization algorithm than can be attained by using alternate color spaces or by fine tuning the histogram difference functions [55].

4.2.1.3 Edge based features

The edges of objects between two adjacent frames in a cut cannot usually be found and appropriately put in correspondence. An edge based feature approach was presented in [70] that used the so-called Edge Change Ratio (ECR) and was further refined in [71]. The ECR is defined as number of dilated edge pixels in two adjacent frames that do not conform. Edges are detected using a Canny [72] edge detector, and in order to handle object motions a dilated edge is compared in a windowed area around the pixels rather than a single pixel. Such a method is prone to failure in the presence very fast camera or object motion, multiple moving objects, moving cameras with moving objects, and occlusions. A comparison of histogram techniques against the edge change ratio technique has shown that the histogram techniques provide similar results without the added complexities [55].

While the ECR methods provide advances in capabilities, especially for fades and dissolves, they suffer greatly increased runtimes due to the added complexities. A recent review [73] managed to get real time capabilities of the edge feature-based method presented in [71] but only on micro-frames of 88x72 pixels. When the frame sizes were in-

creased to 352x288 (a more standard resolution) the frame-processing rate dropped to approximately 2 frames per second.

4.2.2 Classifying Differences as cuts and non-cuts

Once a metric that quantifies the inter-frame differences has been defined, a classifier is required to separate the differences into cuts and non-cuts. Two basic classification techniques that revolve around thresholding techniques as linear discriminators have been proposed; they are global and adaptive thresholding.

4.2.2.1 Global threshold

The input to a global thresholding technique is all of the difference values for a given video, which in the ideal case is supposed to show a single large peak at hard cut locations. A hard cut is declared each time the feature difference value surpasses a globally fixed threshold. A common problem of global thresholding is that in practice it is impossible to find a single global threshold that works with all kinds of video material [55].

4.2.2.2 Adaptive threshold

The input to an adaptive thresholding technique is a windowed subset of difference values for a given video, which in the ideal case is supposed to show a single large peak at cut locations. A hard cut is detected based on the difference of the current feature values with respect to its local neighborhood. Usually a temporal sliding window of size w centered on the current frame is chosen to represent the local neighborhood [55]. A cut is classified when the ratio between the largest and the second largest value in the window surpasses a second threshold [59].

In Figure 4.2, shots are easily seen to be length 2 (664-665) and length 4 (666-669). Both adaptive thresholding techniques in combination with color histogram differences between frames have been shown to lead to higher performance [59, 60]. However this type of adaptive thresholding is prone to false negatives in highly edited sequences. We have found that in commercial video sequences, shots of length two, three and four frames are more common than one would expect. Figure 2 shows a sequence with sev-

eral cuts that may be missed when the window in an adaptive thresholding technique is too large.

FIGURE 4.2: SEQUENCE FROM THE MOVIE PSYCHO[3].

4.3 Feature Tracking for Quantifying Dissimilarity

We propose in this paper a new approach that uses feature tracking as a metric for dissimilarity. Furthermore we propose a methodology to automatically determine a threshold value by performing density estimation on the squared normalized per-frame lost feature count. It has been reported that the core problem with all motion-based features is due to the fact that reliable motion estimation is far more difficult than detecting visual discontinuity, and thus less reliable [55]. Effectively, a simple differencing technique is replaced with a more complex one. Experimentally we have found that the proposed feature tracking method performs flawlessly on all simple[4] examples where pixel and histogram based methods did not achieve such perfect results.

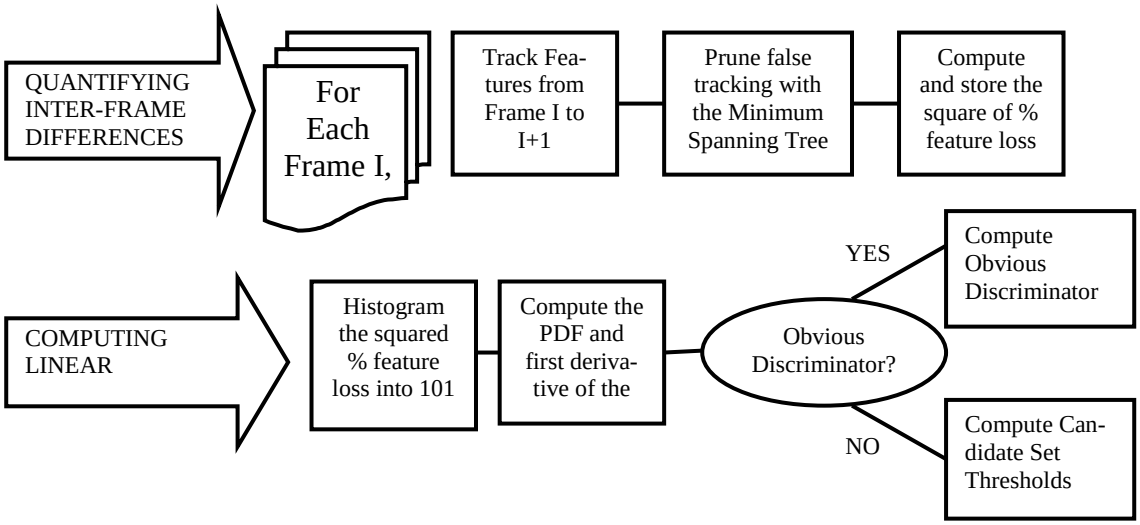


FIGURE 4.3: DIAGRAM OF SYSTEM TO COMPUTE CUTS

We continue by outlining the feature tracking method, an outlier pruning algorithm and a signal separation methodology. We follow up in the next section with a method to dy-

[3] All copyrights © belong to their respective owners. Psycho is an Alfred Hitchcock movie, produced by Shamley Productions and distributed in North America by Universal Pictures.
[4] Here we define simple to be cases of clearly obvious cuts, which were well separated over time and space.

namically select a global threshold. In Figure 4.3 we see the entire flow chart for computing the positions of cuts in a video sequence. Each block within the diagram is detailed in this section and the next.

4.3.1 Feature Tracking

Previous feature based algorithms [70, 71] rely on course-grained features such as edges and do not track edge locations from frame to frame. Rather they rely on sufficient overlap of a dilated edge map and search a very small local area around the original edge locations. In contrast, the proposed method of tracking fine-grained features (corners) on a frame-by-frame basis in less constrained by the original location due to the pyramidal tracking approach. This allows the proposed method to be more robust to object and camera motions. Cuts are detected by examining the number of features successfully tracked (and lost) in adjacent frames, refreshing the feature list for each comparison.

We utilize a corner-based feature tracking mechanism to indicate the characteristics of the video frames over time. As we track corner features over time, we detect production features within the video and can annotate the sequence depending on the features that are successfully tracked over time versus those that are lost. Feature tracking is performed on the luminance channel (grey map) for the video frames. The luminance channel is computed as follows:

$$\text{Luminance} = \text{Red}*0.299 + \text{Green}*0.587 + \text{Blue}*0.114 \tag{4.4}$$

The feature tracker we use is based on the work of Lucas and Kanade in [48]. This work was further developed by Tomasi and Kanade in [49] of which Shi and Tomasi provide a complete description in [50].

Briefly, features are located by examining the minimum eigenvalue of a 2x2 image gradient matrix that is noticeably similar to the Harris corner detector [44]. The features are tracked using a Newton-Raphson method of minimizing the difference between the two windows around the feature points. We continue by presenting a very brief outline of the work by Tomasi et al [48, 49, 50].

Given a point p in an image I, and its corresponding point *q* in an image J, the displacement vector δ between *p* and *q* is best described using an affine motion field:

$$\delta = Dp + t \tag{4.5}$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \tag{4.6}$$

is a deformation matrix and t is the translation vector of the centre point of the tracked feature window. The translation vector t is measured with respect to the feature in question. Tracking feature *p* to feature *q* is simply the problem of determining the six parameters that comprise the deformation matrix D and the translation vector t. In the case of pure translation, D will be the identity matrix and thus

$$\delta = p + t \tag{4.7}$$

Because of this, the case of pure translation is computationally simpler and thus preferable. Since the motion between adjacent frames of standard video is generally quite small, it turns out that setting the deformation matrix to identity is a safe computation [50], leaving us with the translation vector being exactly the displacement vector. Complete details for the tracking equations and feature selection can be found in Chapter 2, Section 11.

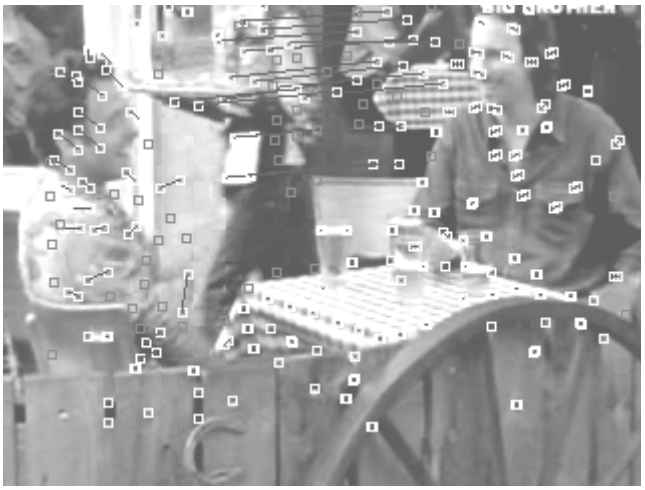


FIGURE 4.4**:** END RESULT FOR FEATURE TRACKING OVER SEVERAL FRAMES.

In Figure 4.4, stationary objects in the foreground and a quickly moving object (noted by the long motion vectors) in the background. Gray squares are lost features; white squares represent the tracked feature and its original position.

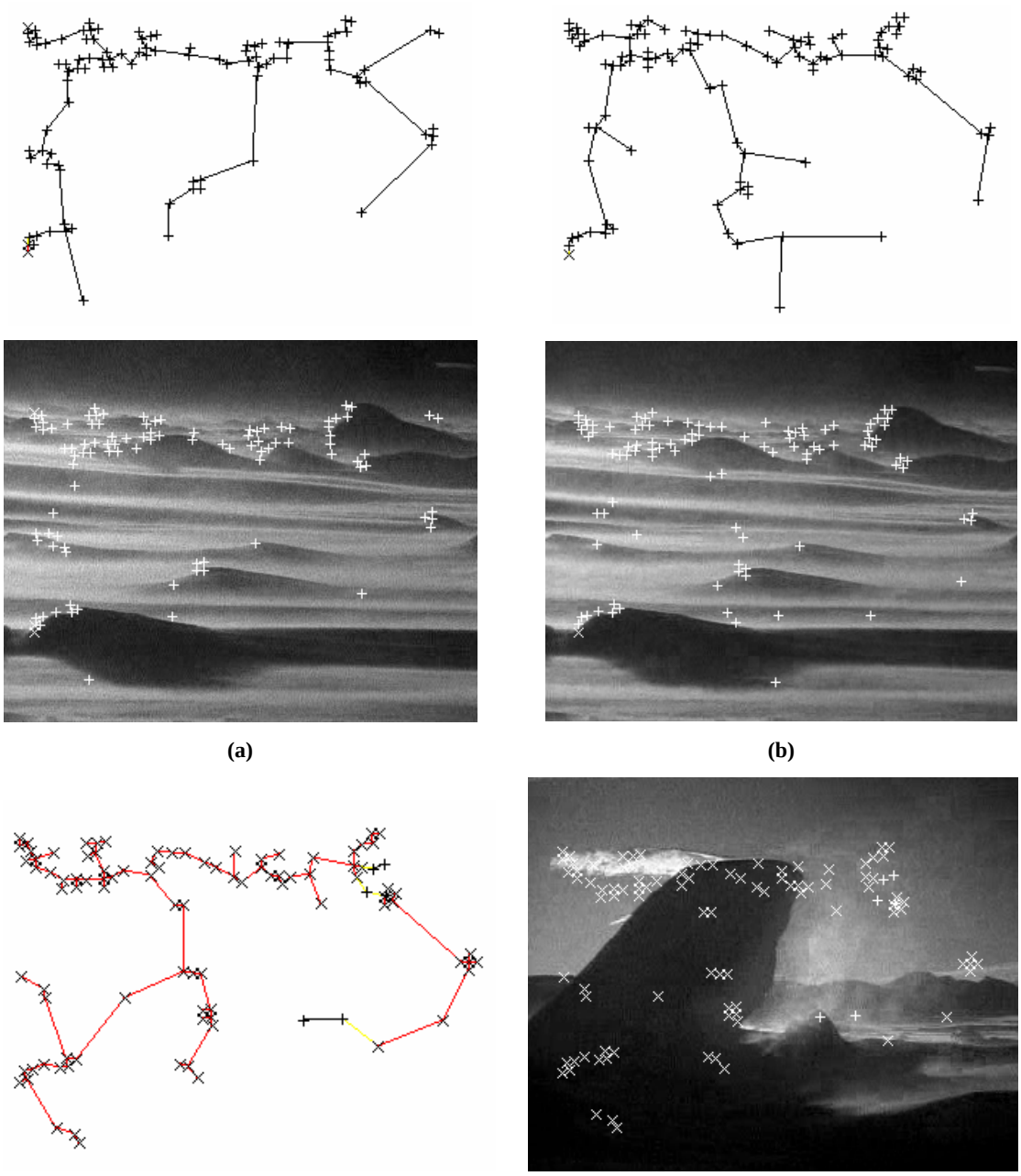
The displacement vector is computed using a pyramid of resolutions because processing a high resolution image is computationally intensive. The multi-resolution pyramid within the feature tracker reduces the resolution of the entire image, say by a factor of 2. Tracking occurs by tracking a features general area in the lowest resolution and upgrading the search for the exact location as it progresses back up the pyramid to the highest resolution. An example of tracked feature and displacement vectors is given in Figure 4.4.

While tracking features it is possible that an extremely large object motion between frames does occur. It has been noticed that in such cases the tracking mechanism begins to fail because the disparity between adjacent frames is too large. The result, features are lost and cannot be tracked any further. This fact indicates that some large shift in the adjacent frames has occurred and can be handled at the cost of substantially higher processing time by increasing the pyramid dimensions or by removing the identity constraints of the matrix *D*.

4.3.2 Pruning False Tracking

In the case of a cut at frame i, all features being tracked should be lost from frame i to i+1. However, there are often cases where the pixel areas in the new frame coincidentally match features that are being tracked. In order to prune these coincidental matches, we examine the minimum spanning tree of the tracked and lost feature sets. We can see from Figure 4.5 a, in the case of a cut, that there is a very small percentage of features that are tracked. This is clearly an erroneous situation because the two consecutive frames are so obviously different. We can remove some of these erroneous matches by examining properties of the minimum spanning tree of the tracked and lost feature sets. By severing edges that link tracked features to lost features we end up with several disconnected components within the graph. Any node (feature) in the graph that becomes a singleton (not connected to any other feature) has its status changed from tracked

to lost, and is subsequently included in the lost feature count. The property we are exploiting here is the fact that erroneously tracked features will be minimal and surrounded by lost features. Clusters of tracked (or lost) features have localized support that we use to lend weight to our assessment of erroneous tracking



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

FIGURE 4.5: THREE CONSECUTIVE FRAMES FROM A SEQUENCE.
(a) shows a very high proportion of successfully tracked features from the previous frame to current frame (b) shows successfully tracked features from (a) (previous) to (b) (current) (c) shows those features cannot be found in very high proportion indicating a cut. Above each frame is the minimum spanning tree for each of the feature sets, (+) are tracked features, (X) are lost features.

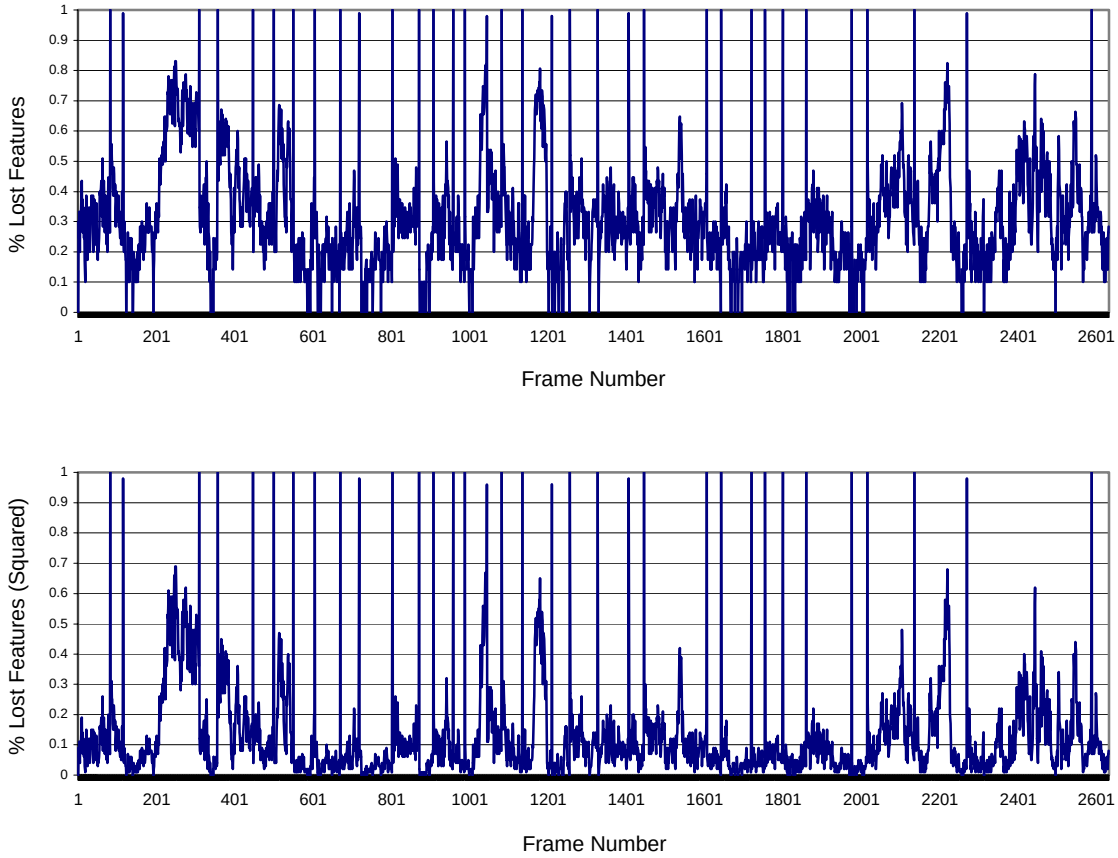


FIGURE 4.6: THE RESULTS OF SQUARING.
Top is the original signal, bottom is the squared signal. The separation between the cuts and the non-cuts has been greatly increased.

Our inter-frame difference metric is the percentage of lost features from frames I to i+1. This corresponds to changes in the minimum spanning tree, but is computationally efficient. Because we are looking to automatically define a linear discriminator between the cut set and the non-cut set, it is advantageous to separate these point sets as much as possible. In order to further separate the cut set from the non-cut set, we square the percent feature loss which falls in the range [0..1]. This has a beneficial property of ensuring the densities of the cut set and the non-cut set are further separated and thus ease the computation of a discriminating threshold. The idea here is that in the case of optimal feature tracking, non-cut frame pairs score 1 (all features tracked) and cut frame pairs score 0, no features tracked. Squaring, in the optimal case, has no effect as we are already maximally separated. However, in practice, squaring forces the normalized values

for non-cut frame pairs closer to zero. Figure 4.6 shows the effect of stretching on the inter-frame differences.

4.4 Automatically Determining a Linear Discriminator

Having a difference metric and a method to further separate the cut set from the non-cut set, we can now compute the linear discriminator for the two sets. There is no common threshold[5] that works for all types of video. Next we present an algorithm to auto-select a global threshold. There are two classes of frame differences, cuts and non-cuts; and our goal is to find the best linear discriminator that maximizes the overall accuracy of the system. The cut set and the non-cut set can be considered to be two separate distributions that should not overlap, however in practice they often do, as illustrated in Figure 7b. When the two distributions overlap a single threshold results in false positives and false negatives. An optimal differencing metric would ensure that these two distributions do not overlap; in such a case the discriminating function is obvious and accuracy is perfect. The quality of the difference metric directly affects the degree to which the two distributions overlap, if any. Until an optimal difference metric is proposed, the problem of optimal determination of the discriminator must be considered.

To avoid the problems illustrated in Figure 2 that may occur with a windowed adaptive threshold, we have opted to examine the density of the recorded inter-frame difference values for an entire sequence. The idea here is that there should be two distinct high-density areas, those where tracking succeeded (Low feature loss) and those where tracking failed (high feature loss). In practice, this situation appeared about 50% of the time in our data set. We will introduce the idea of a candidate set in section 4.4, which is the set of features that can be discriminated by zero crossings of the probability density function that characterizes the densities of the inter-frame differences. It needs to be noted here that while we examine the density for the entire sequence to determine a global threshold, it is possible to apply the method outlined next in a windowed manner to determine localized thresholds.

[5] Note that a threshold is a linear discriminator with the function y = some_value.

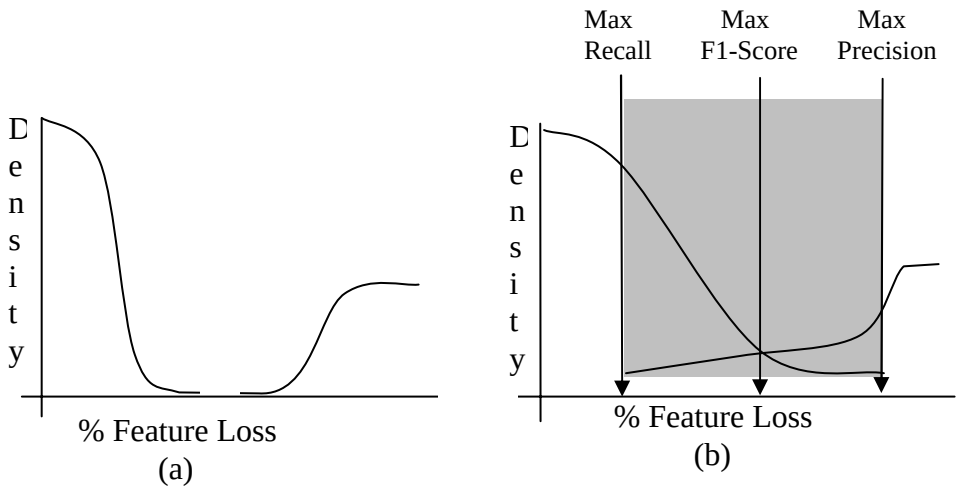


FIGURE 4.7**:** (A) NON-OVERLAPPING DISTRIBUTIONS, (B) OVERLAPPING DISTRIBUTIONS
(a) the discriminator is obvious. (b) the cut set on the right and the non-cut set on the left. The ideal discriminator lies within the overlap region.

4.4.1 Density Estimation

In order to auto-select a threshold, we examine the frequency of high and low feature loss. We are looking to exploit the fact that the ratio of non-cuts to cuts will be high, and therefore the ratio of low feature loss frame pairs to high feature loss frame pairs will also be high. As the frame to frame tracking of features is independent of all other video frames, we have n independent observations from an n+1 frame video sequence. The extrema of the probability density function can be used to determine the threshold to use. We can use the statistical foundations of density estimation to estimate this function.

The intention of density estimation is to approximate the probability density function $f(\bullet)$ of a random variable X. Given that we have n independent observations $x_1, \ldots, x_n$ (our tracked feature percentage squared) from the random variable X (our video sequence). The kernel density estimator for the estimation of the density value $f(x)$ at point x is defined as

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x_i - x}{h}\right) \tag{4.8}$$

Where $K(\bullet)$ is the so-called kernel function, h is the bandwidth (window size), and n is the number of samples (number of frames-1). There have been variety of kernel func-

tions presented in the past and we performed an empirical evaluation of the 9 kernel functions listed in Table 1 to determine which kernel is the most appropriate for our problem.

Table 1: Density Estimation Kernel functions.

Kernel Name	Kernel Function K(α)
Uniform	½ α
Rectangle	α
Epanechnikov	¾ $(1 - \alpha^2)$
Biweight	15/16 $(1 - \alpha^2)^2$
Triweight	35/32 $(1 - \alpha^2)^3$
Gaussian	$\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\alpha^2}$
Triangular	$\lvert \alpha \rvert$)
Cosine Trace	$\frac{\pi}{4}\cos\left(\frac{\pi}{2}\alpha\right)$
Laplacian	$\frac{1}{2}e^{\lvert\alpha\rvert}$

We want a kernel estimator that will facilitate the identification of extrema in the probability density function. In Figure 8 we examine each of the kernels in detail to evaluate the effects the smoothing kernel has on each. It is important to select a kernel that does not over smooth, resulting in a loss of discrimination capabilities. As well, we must not select a kernel that under-smoothes, resulting in a ragged signal and thus a misrepresentation of the position of extrema. In the case where the distributions overlap, we determined that the triangular kernel provided the best smoothing properties.

Generally speaking, we found that the Laplacian, Uniform and Rectangular kernels under smoothed the signal, leaving too many extrema for reliable subsequent analysis. We also found that the Gaussian, Triweight, and Epanechnikov kernels over smoothed the signal, making accurate determination of extrema difficult. The remaining 3 kernels, Biweight, Cosine Trace and Triangular appeared to have a very similar effect on the original signals. The triangular kernel was selected from the remaining three because it did not over-smooth locally, making the determination of extrema easiest of the three. In Figure 4.8 we see an analysis of the all kernels for a single data source with non-overlapping distributions. The results presented were consistent across many different samples from our data set. From Figure 8 we draw the conclusion that in the case that

the two distributions do not overlap, a smoothing of the function will not destroy the obvious discriminating threshold.

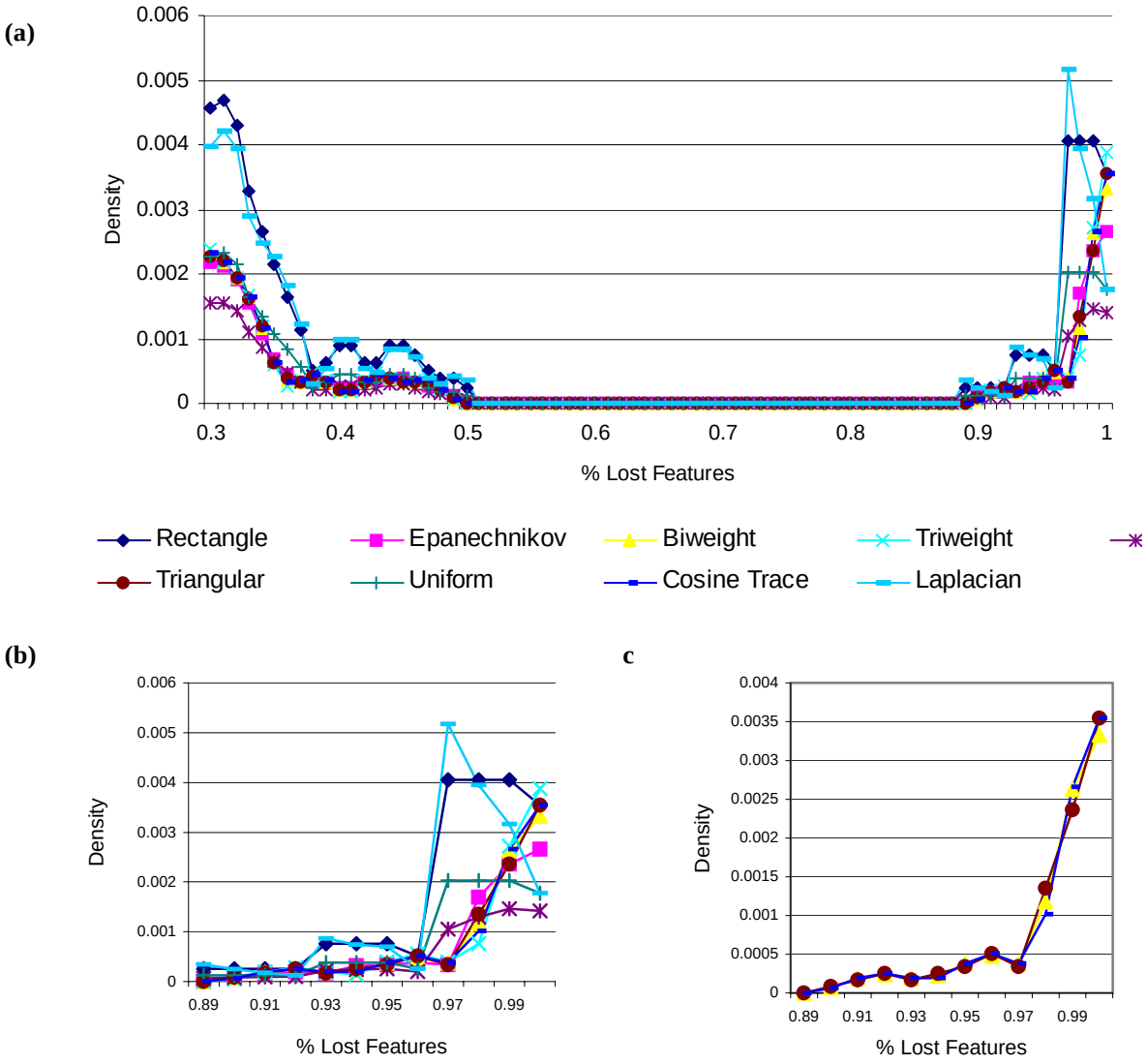


FIGURE 4.8: DETAILED EXAMINATION OF KERNEL FUNCTIONS.
(a) Examination of all 9 kernels for the same video sequence. (b) all nine kernels at the high feature loss areas. (c) Biweight, Cosine Trace, and Triangular kernels

Now that the triangular kernel has been selected to smooth our function, we need to determine the bandwidth (window size) for the kernel. We performed a analysis with kernel widths 3,5,7,9,11,13,and 15. In Figure 9 we show a triangular kernel for window sizes 3,7,11 and 15. Widths 9 through 15 represented almost identical curves and thus any kernel width over 7 provided no further information and likely is over smoothing.

The apparent extrema that appeared in widths 3 and 5 indicated under smoothing. By elimination, we were left with a kernel width of 7, which has provided good results in our experiments.

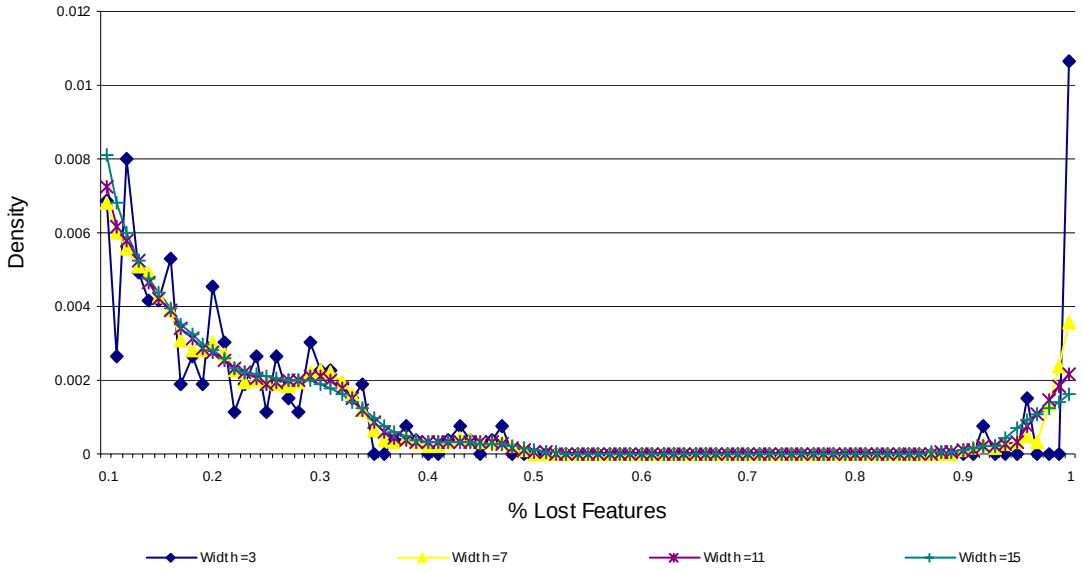


FIGURE 4.9: TRIANGULAR KERNEL AT VARIOUS BANDWIDTHS (WINDOW SIZES)

4.4.2 Computational Considerations

In the case of an exact computation of the density estimates, the kernel function must be evaluated $O(hn^2)$ times. This increases the computation time as the number of frames becomes large (keep in mind that a 2 hour movie contains 216,000 frames). An alternative, and faster, way is to approximate the kernel density estimate is to use the WARPing (Weighted Average of Rounded Points) method [15] The core concept behind WARPing is to effectively histogram the data into bins of length *d*. The bin centre of its corresponding bin then replaces each observation point for subsequent computation. A typical choice for *d* is to use *h*/5 or $(x_{max}-x_{min})/100$. In the latter case, the effective sample size i can be at most 101. This property nicely reflects our situation, where we are keeping track of the percentage of features tracked per frame pair, which is in the range of 0 to 100 percent, or 101 bins.

For the WARPing method, the kernel function needs to be evaluated only at $O(h \cdot r/d)$ plus the initial pass to histogram the data being $O(n)$. In total, the number of steps is $O(n)$ +

O(h·r/d) where h is our window width (7), the range *I* is 101 and the bin length is 1. This reduces the number of steps to *O(n) + O(h·r)*. Since *n* greatly exceeds *h·r* (707 in our experiments), we are have an upper bound of *O(n)*. This is considerably faster than the exact computation, when the sample size is large.

4.4.3 Non-Overlapping Distributions

Non-overlapping sets of distributions are very easily determined by looking for a large plateau of zero density. The first appearance (traveling the curve from 0 to 100) of a large plateau indicates the range of the separation point. Selecting the extreme end point (closest to the non-cut set) for the threshold has yielded the correct result on all cases of non-overlapping distributions in our test suite.

4.4.4 The Candidate Sets (Overlapping Distributions)

We now introduce the ideas around what we term the candidate sets. We define 3 candidate sets, where each set contains the frames that maximize the precision, F1 and recall rates. Precision is the portion of the declared cuts that were correct. Recall is the portion of the cuts that were declared correctly. F1 is a combination of precision and recall. A complete description of these terms and their formulae are given in the experiments section. Depending on user need, precision, recall or best overall performance, these candidate set thresholds are now able to be determined.

The candidate sets are 3 sets that for convenience we will call the precision set (*P*), the F1 set (*F*) and the recall set I. These sets have the following property:

- $R \subseteq F \subseteq P$

In the case of non overlapping sets, precision, recall and F1 scores are all 1.0 and the frames in each set are the same. In the case of overlapping sets, the frames in the precision set appear in the F1 set, and those in the F1 set appear in the recall set.

The candidate sets are determined by examining zero crossings of the first derivative of the computed probability density function (PDF). There are often many consecutive zero crossings of the function over time, so we use a modified function G(x) to make

the large changes in density more apparent. The first derivative of the PDF *f(x)*, is modified to a function G(x) using the following rules:

$$\begin{aligned} G(x) &= g(x) + g(x+1): \\ &\text{if } \hat{f}'(x) \le 0 \text{ then } g(x) = 0 \\ &\text{if } \hat{f}'(x) > 0 \text{ then } g(x) = 1 \end{aligned} \tag{4.9}$$

In Figure 10, we see the original first derivative function and the modified function.

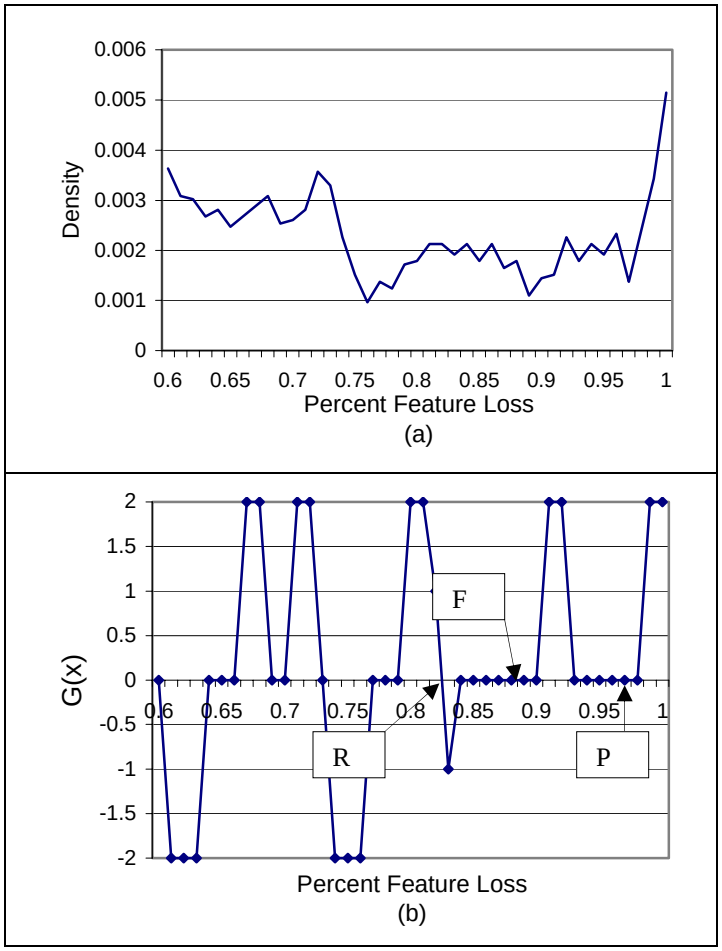


FIGURE 4.10: (A) ORIGINAL PDF (B) MODIFIED FIRST DERIVATIVE (G(X))

The zero crossings, starting from 1.0 and following G(x) as x decreases are used to determine the thresholds for each of these sets using the criteria listed here:

- (*P*) is The first zero crossing
- (*F*) The position of the minimum of PDF corresponding to the plateau of G(x) given:

  If the next zero crossing has opposing direction as the first (i.e. is not u or n shaped) and is part of the plateau of first zero crossing use this plateau, otherwise use the next plateau.
- The next subsequent zero crossing

The arrows in Figure 4.10(b) point to the zero crossings used. The first zero crossing is at 0.98 (P) and because the next zero crossing at 0.93 is also an upwards direction (u shaped), we skip to the next plateau to determine F. The next zero crossing (not on the plateau) is used for R.

4.5 Experiments

In this section we perform a variety of experiments on data sources that were deemed to be difficult[6]. We outline our metric for comparing the proposed method against other methods followed by the experimental results. We conclude this section with some information on running time and how feature count and selection will affect the system.

4.5.1 Comparison Metrics

Contingency tables are often used in the quantification of the results categorization systems. Considering the system to be a 2-category classifier (cuts and non-cuts) we can do the evaluation of the effectiveness using a contingency table and its associated statistics. Commonly, precision and recall are used, however we will also address accuracy, error and the so-called F1 score as means of evaluating the effectiveness of the cut detection.

We continue by defining each statistic available for use in our evaluation. All of the statistics are calculated based on a so-called contingency table, where our classifier (cut detector) detects cuts (positives) or non-cuts (negatives). The cut detector can properly detect a cut (true positive), improperly detect a cut (false positive), properly detect a non-cut (true negative), or improperly detect a non-cut (false negative). The elements in the set of cuts and non-cuts will never intersect because a frame cannot be double classified as both a cut and a non-cut. Our contingency tables have this form:

	True Cut	True Non-Cut
Classified Cut	True Positive (T^+)	False Positive (F^+)
Classified Non-Cut	False Negative (F^-)	True Negative (T^-)

[6] By difficult we mean frames with quick motion, both camera and object, many cuts in short succession.

T^+, F^+, F^-, and T^- are the counts that reflect how the classified categories matched the correct categories. From the contingency table we can compute the following statistics:

Accuracy – This measures the percentage of all decisions that were correct decisions. Range: 0 to 1, with 1 being the best score. It is defined as:

$$Accuracy = \frac{T^+ + T^-}{T^+ + T^- + F^+ + F^-} \tag{4.10}$$

Error – This measures the percentage of all decisions that were incorrect decisions. Range: 0 to 1, with 1 being the best score. It is defined as:

$$Error = \frac{F^+ + F^-}{T^+ + T^- + F^+ + F^-} \text{ or: 1-Accuracy} \tag{4.11}$$

Precision – This measures the percentage of the classified categories that were correct. Range: 0 to 1, with 1 being the best score. It is defined as:

$$Precision = \frac{T^+}{T^+ + F^+} \tag{4.12}$$

Recall – This measures the percentage of the correct categories that were classified. Range: 0 to 1, with 1 being the best score. It is defined as:

$$Recall = \frac{T^+}{T^+ + F^-} \tag{4.13}$$

F1-score – This measures a combination of precision and recall. Range: 0 to 1, with 1 being the best score. It is defined as:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.14}$$

In terms of T^+, F^+, and F^-:

$$F1 = \frac{2 \times T^+}{2 \times T^+ + F^+ + F^-} \tag{4.15}$$

The F1 score is the only statistic that is worth trying to maximize on its own. Perfect precision can be achieved by never detecting a cut and perfect Recall by always detecting a cut. A truly accurate system will assign the correct categories and only the correct categories, maximizing precision and recall at the same time, and therefore maximizing the F1 score.

Attempting to maximize the accuracy score, thus minimizing error, is an inappropriate measure in the case because the size of the non-cut set is so large in comparison to the cut set. Simply declaring all frames as non-cuts will result in a high accuracy. Any metric for cut detection that uses the True negative number in its evaluation is not a good indicator of quality because of the distribution of cuts to non-cuts in video data. This is casually confirmed if we consider the redundancy that video data contains.

In [28], the authors suggest an alternate computation for accuracy as:

$$\text{Accuracy} = 1 - \frac{F^-}{T^+ + F^-} - \frac{F^+}{T^+ + F^+} \tag{4.17}$$

With some algebraic manipulation, we can see their definition of accuracy is simply

$$\text{Accuracy} = \text{Precision} + \text{Recall} - 1 \tag{4.18}$$

As Matter and Robinson's metric is a function of precision and recall, we will omit using it and rely on the more standard metrics of precision, recall and the F1 score.

4.5.2 Experimental Results

To start, we perform a set of experiments to compare the proposed method against a histogram-based method, specifically 'cutdet' from the MOCA project [27]. We used the precompiled version of cutdet and treated the internals as a black box. We ran a sequence through cutdet with various threshold settings to determine its characteristics. We examined the precision, recall, and F1 score. The sequence is from a television show and the quality of the capture is quite high. The action and motion is not extreme, and the colours are vibrant and distinct. It was assumed, based on the ideas behind histogram comparisons, that this sample would highlight the capabilities of the cutdet system. In Figure 11 we see that the F1 score has platitude at threshold 0.45 which indicates the best threshold for cutdet on this particular sequence. The exact values of precision, recall and F1 score are: 1, 0.941, and 0.969 respectively. In the case of perfect detection, precision, recall and the F1 score will all be 1. As the F1 score hit a platitude at 0.969, the cutdet method would be unable to achieve a perfect score in this example.

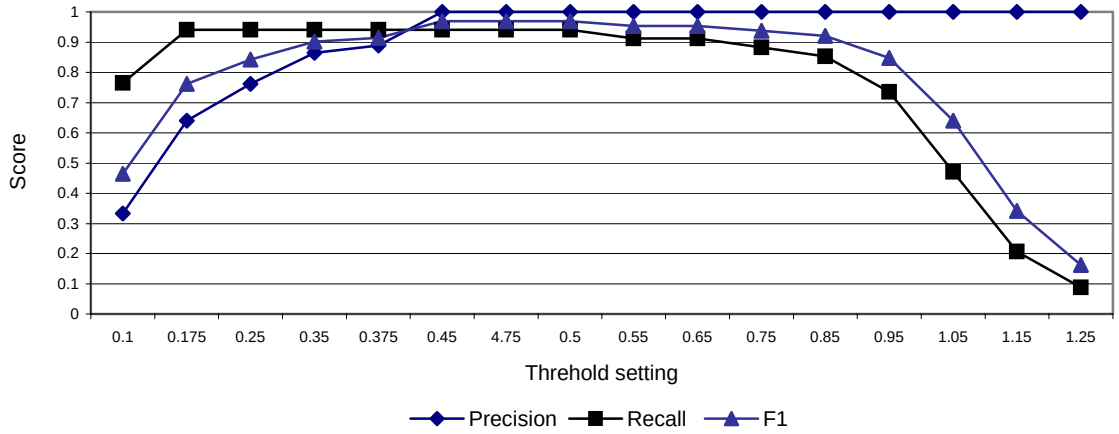


FIGURE 4.11: GRAPH OF PRECISION, RECALL AND F1 SCORES FOR A SEQUENCE RUN THROUGH THE CUTDET SYSTEM.

When the proposed method was run against the same sequence, it received a perfect detection rate. The clear selection of threshold can be seen in Figure 12 as the feature tracking was clearly working very well and the separation between the cut set the non-cut set is obvious due the large space of zero density between the cut and non-cut sets.

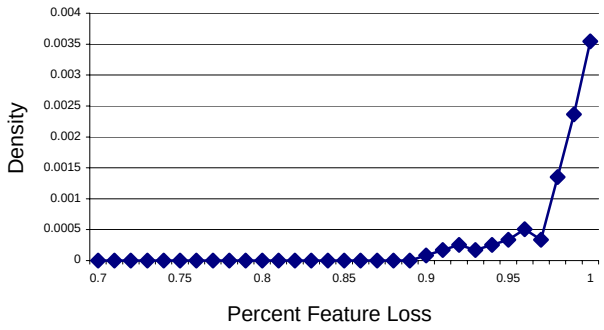


FIGURE 4.12: DENSITY ESTIMATION GRAPH.

In the experiment that follows, a selection of video clips that represent a variety of different video genres are used for cut detection. In Table 2, we present the data set with explanation why it is being used in the experiments. We compare the results of the proposed method against a pixel-based method with localization information and a histogram based method (specifically cutdet, from the MOCA project [27]). The localization information in the pixel based method relies on the statistical improbability that the positions of pixel values will remain the same over a cut frame pair. In this experiment, we are attempting to maximize the F1 score. For the proposed method, we ran each sample

through the system once and computed the F1 candidate set threshold. For cutdet, and the pixel based method we ran 13 different thresholds and computed the precision, recall and F1 scores for each of the runs. The best F1 score was selected for comparison. This effectively required the two methods to be executed 13 times in comparison to the 1 time for the proposed method.

Table 2: Experimental data set

Source Label	**Characteristics of video data**	**Genre**
A	Cartoon clip. Substantial object motion.	Cartoon
B	Substantial object motion. This clip is taken from a film where a blue filter was used to simulate low lighting conditions.	Action
C	Black and white movie. Substantial action and motion. Many close proximity cuts. This clip is the murder scene from the movie psycho.	Horror
D	High quality □pipolar□ct□ of a television show.	Drama
E	Low quality □pipolar□ct□ of a television show.	Sci-Fi
F	Commercial, no cuts, quick motion, many production effects. Meant to show that dissolves are not mistakenly classified as cuts.	Commercial
G	Commercial sequence from the MOCA Project	Commercial
Q	Video abstract from the MOCA Project	Comedy/Drama
I	News Sequence from the MOCA Project	News/Documentary
J	Trailer for a film. This clip has many computer generated features, many close proximity cuts. Trailer for the movie Lawnmower Man.	Trailer/Sci Fi/Action

In Table 3, we present the results of running the 3 methods on the dataset. The proposed method outperforms both the histogram-based method and the pixel based methods. In most cases (8 of 10) the proposed method provides the maximal F1 score. A simple statistical analysis of the overall capabilities is given at the end of Table 3. The average, variance and standard deviation for the 10 samples were computed. On average, the proposed method significantly outperforms the other two methods. The variance and the standard deviation show that the results offered by the proposed method are also more stable across a variety of different video genre.

Table 3: Results on data set.

	Proposed feature tracking method			**Pixel Based method with localization**			**Histogram MethodCut Det (MOCA)**		
Data Source	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
A	1	1	1	1	1	1	1	1	1
B	1	1	1	.825	.825	.825	1	.375	.545
C	.595	.870	.707	.764	.778	.771	.936	.536	.682
D	1	1	1	1	1	1	1	.941	.969
E	.938	1	.968	.867	.867	.867	.955	.700	.808
F	1	1	1	0	0	0	1	1	1
G	.810	.944	.872	.708	.994	.809	1	.667	.800
H	.895	.895	.895	.927	1	.962	.971	.895	.932
I	1	1	1	1	1	1	1	.500	.667
J	.497	.897	.637	.623	.540	.591	.850	.395	.540
AVG	**.874**	**.961**	**.908**	**.774**	**.800**	**.783**	**.971**	**.701**	**.794**
VAR	**.034**	**.003**	**.018**	**.090**	**.101**	**.093**	**.002**	**.060**	**.036**
STD. DEV	**.185**	**.054**	**.134**	**.301**	**.318**	**.304**	**.048**	**.246**	**.190**

It is not surprising that ‘cutdet’ out performs the proposed system in H, because the abstract was created by the MOCA project from which cutdet originates. However, it is surprising that the pixel based method outperformed both. In examples C and J, the F1 score was not maximized as the heuristic to determine the F1 candidate set threshold did not achieve the best value, rather a good value. Within the range of the F1 candidate set threshold plateau, maximum F1 was achievable.

4.5.3 Processing Speed

For all the experiments given in Table 3, we tracked 100 features with a minimum inter-feature distance of five pixels. The processing time for a frame pair is approximately 70 milliseconds on a 2.2 GHz Intel processor on frames sized 320x240. This is significantly faster than [28] without a loss in granularity. Unlike the pixel and histogram based methods, the running time of the tracker is not a function of the size of the video frames and remains constant regardless of the video size. Tracking over an n-frame interval can cut this time by a factor of n. By skipping every other frame, we still maintain the 70ms processing time per frame pair, but have reduced the number of frames by a

factor of 2. By increasing the number of frames skipped, we decrease the frame accuracy to which we can report, and risk losing tracking context in the presences of large object and camera motions. We ran several data samples through the system set to skip one and two frames between tracking. As expected the running times reduced to approximately half and a third respectively. However, we experienced a reduction in the F1 scores in half the samples and all the samples for skipping one and two frames respectively. Playing with the tracking parameters may reduce the error, but is beyond the scope of this work.

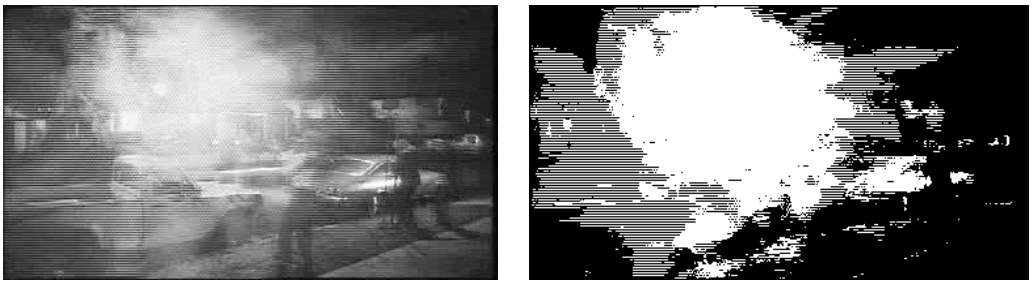
4.5.4 The Effect of Feature Selection

We have chosen the number of features to track to be 100. However it should be noted that the optimal number of features to track has yet to be determined, if at all possible. Logically, there must be enough features to track and ensure a certain amount of coverage of the frame, however selecting too many features will result in features that are less than ideal being selected for tracking and therefore increase the likelihood that those features will not be reliably tracked from frame to frame. We have performed some rudimentary experiments that show as the number of features selected to be tracked decreases, and thus the overall quality of those features (for tracking purposes) increases, the percentage of lost features in a cut situation increases overall. Specifically, when the number of features tracked decreases, the threshold that is automatically selected approaches 100 on simple sequences. However, when the number of features tracked decreases, the system becomes more susceptible to object motion, occlusions and very quick camera motion because we lack spatial coverage of the frame, therefore reducing the accuracy on more difficult sequences.

Another selection option that needs consideration is how wide the non-maxima suppression window should be. This option is effectively selecting the best corner feature within a given radius. By selecting a radius that is too large will result in features being selected that are less optimal for tracking while a radius that is too small results in feature clustering that is prone to loss due to object motion and occlusions. We have found empirically that a radius of five pixels provides good results; a radius of fifteen pixels provides worse results.

4.5.5 Known Problem Areas

There are some very clear restrictions on this feature-based method for detecting cuts. Primarily, the quality of the digital video plays a fundamental role because we are tracking fine-grained features (corners) rather than coarse-grained features such as edges, color blobs, and histograms. Overly noisy digitization will result in the feature tracking having a difficult time properly tracking selected features and will result in a higher overall percentage loss of tracked features. This will result in an overlapping of the cut set distribution and the non-cut set distribution and therefore result in an overall lower accuracy. As well, dropped frames in the digitization process may result in unnatural spaces between frames. This will also cause the feature tracker to potentially lose features and categorize a cut. To complicate the matter, it is not clearly defined whether or not these situations should be classified as cuts. Other anomalous glitches in the digitization process will result in feature loss as well. The figure below illustrates digitization artifacts that can cause problems for our fine-grained feature tracker because of the high gradient changes on the digitization scan lines.

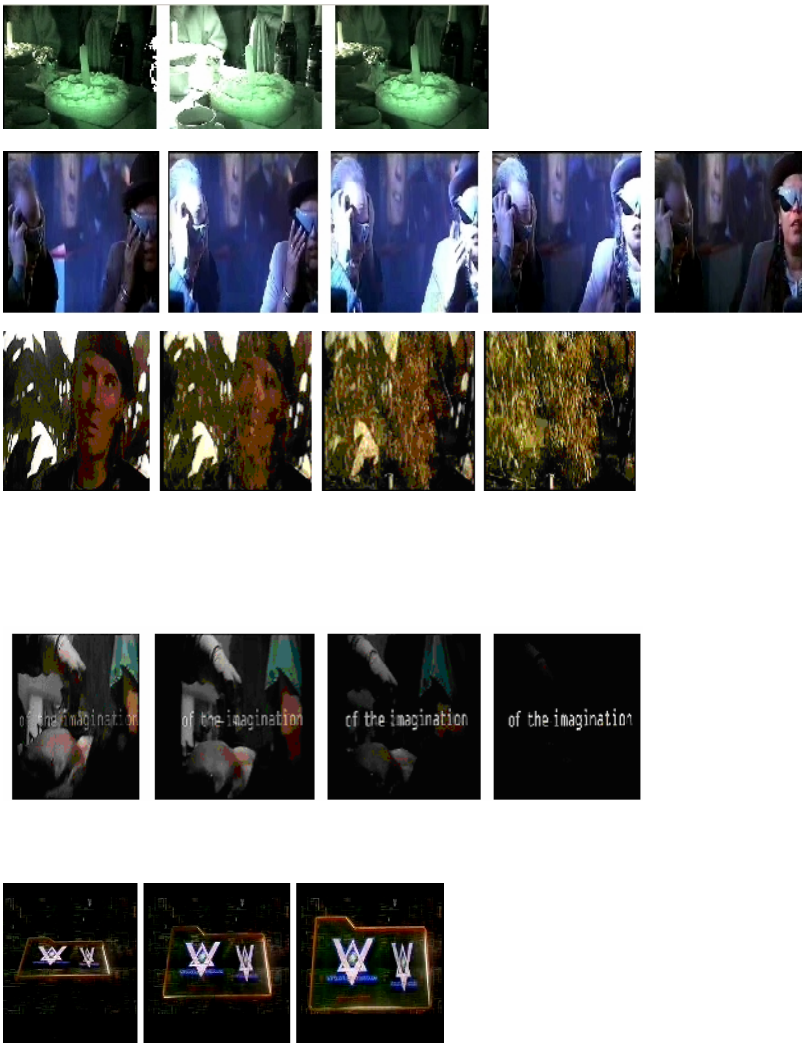


(a) (b)

FIGURE 4.14: DIGITIZATION ARTIFACTS
(a) original frame (b) exemplary amplification of artifact errors. Notice the lines around the explosion.

Each of the examples in Figure 15 represents problems that the proposed cut detection method faces. Because we are tracking features in the luminance space, we are subject to feature loss when large changes in brightness occur. The system is robust to gradual dissolves, however high speed dissolves (over a small number of frames) cause problems because of the large the number of pixels that change greatly. This results in the tracking windows being very different and the correlation computations result in residuals that are simply too large. Finally, we have noticed in several cases that problems

occur with computer-generated video data. This is usually due to the fact that the objects in computer generated video move at rates that are unusually quick. Take for example the 3 consecutive frames from Figure 15, part e: The motion exceeds what a normal object videotaped at 30 frames per seconds (often because animation is at 10 frames per second), and failure occurs.



a) Sudden flash, causing an abrupt change in luminance, improperly declared a cut.

b) A longer sequence of changed illumination, returning to original illumination.

c) A quick dissolve. Changes in luminance declared as cuts, or fade to black because the large change in pixel values caused the tracking systems correlation checks to fail.

d) A quick dissolve over two frames. Often declared as two consecutive cuts because the pixel changes cause the tracking systems residual checks to fail.

e) Computer generated graphics that move sharply. There is also little texture allowing inferior selection of features for tracking.

FIGURE 4.15: EXAMPLES[7] OF PROBLEM SITUATIONS.

[7] All copyrights © belong to their respective owners. Lawnmower Man is a Brett Leonard movie, produced and by distributed in North America by New Line Cinemas.

4.6 Conclusions and Discussion

We have presented a fine-grained feature-based method for video segmentation, specifically cut detection. By utilizing feature tracking and an automatic threshold computation technique, we were able to achieve F1, recall and precision rates that generally match or exceed current methods for detecting cuts. The method provides significant improvement in speed over other feature-based methods and significant improvement in classification capabilities over other methods. The application of feature tracking to video segmentation is a novel approach to detecting cuts.

Due to problems associated with a window based adaptive thresholding, we have introduced the concept of candidate sets that allow the user to prejudice the system towards results that are suitable to individual needs. This kind of thresholding is a novel approach to handling the overlapping region of two distributions, namely the cut set and the non-cut set in video segmentation.

Chapter 5

Autocalibration from the Fundamental Matrix

5.1 Introduction

Calibration is the process of computing internal physical quantities of a camera's geometry. Parameters such as focal length, center of projection, and CCD sensor array dimensions are required in order to get 3D information from a series of images. Autocalibration has become popular recently because of the desire to create 3D reconstructions from a sequence of uncalibrated images without having to rely on a formal calibration process. The standard calibration model for a pinhole camera has five unknown intrinsic parameters defined in a 3x3 calibration matrix (K). These parameters are the focal length, aspect ratio, sensor skew and the center of projection x and y (the principal point). The accurate estimation of these 5 parameters directly from an image sequence without having a formal calibration process is the ultimate goal of autocalibration.

Autocalibration works by computing aforementioned quantities directly from 2D image correspondences, and then using invariants of these quantities to find the camera calibration. The fundamental matrix, and the full projective reconstruction are two quantities that can be computed from a set of 2D image correspondences, and they are the basis of most autocalibration algorithms. As such autocalibration algorithms can be divided into three classes that we will refer to as classes A, B and C. In class A algorithms, we compute the calibration matrix K from the fundamental matrix (the recovered epipolar geometry) [75, 76, 77, 78, 79]. In Class B algorithms (K) is computed from a projective reconstruction [17, 80, 81] of the scene. Class C algorithms autocalibrate from homographies and planar features within an image sequence [82, 83].

While Class C algorithms can compute intrinsic camera parameters from a set of inter-image homographies [84], we loosely consider them autocalibration routines. Because a homography is a planar transformation, Class C algorithms require the use of planar targets [85, 103] or the automatic detection and correspondence of planar regions

within an image sequence. While it has been shown that planar regions may be robustly detected in images [52], it is highly probable that an image sequence will exist where there are no planar objects, or the existing planar objects are not suitable for robust detection. The aforementioned requirements must be known apriori to the computing the calibration parameters and therefore Class C algorithms are not generalized, rather they rely on specific features that may not be present. Due to these facts, it is questionable whether or not a Class C algorithm is truly an autocalibration routine in the sense that it requires a target (therefore not autocalibration), or is presupposed by the planer region detection/correspondence problem (therefore not generalized). Because of these problems, Class C algorithms are not considered in this work.

In this work we compare against class B algorithms with are thought to be numerically superior to other calibration methods. Since the projectively reconstructed frames must all be warped to a consistent relative base, Class B algorithms are computationally difficult in comparison to simply finding the fundamental matrix between image pairs. It is often claimed that Class B autocalibration algorithms are superior to Class A and Class C algorithms because those algorithms do not enforce the constraint that the plane at infinity (an invariant between projective and Euclidean space) be the same over the entire image sequence [86]. It is precisely this constraint that makes Class B algorithms computationally difficult. In this paper, we provide evidence that Class A algorithms combined with the use of evolutionary systems produce as accurate an autocalibration as their Class B counterparts.

Another concern with Class A algorithms is the existence of extra degenerate motions, these being pure rotations, pure translations, affine viewing and spherical camera motions [86, 87]. However, there exist many practical situations that do not contain these degenerate motions. Also, in many cases autocalibration is the only option, and even a less accurate autocalibration result is better than no calibration at all. For example, there are many photographs and video clips in existence for which there is no knowledge of the camera. In order to reconstruct the 3D world from those image sequences, autocalibration is the only option.

Autocalibration has been criticized in the past [88] because many different calibrations will provide a 3D reconstruction with reasonable Euclidean structure. In other words, the corresponding reconstruction will usually look good because the different right angles look square and the different length-ratios look correct. However, this depends considerably on the image sequence, and the camera used to acquire that sequence. All that we can conclude from this fact is that using the "look" of a reconstruction to evaluate the autocalibration results is unreasonable. It is necessary to have the ground truth camera calibration to do a proper performance evaluation. In this paper we evaluate the proposed autocalibration algorithms on image sequences for which the ground truth camera calibration is known a-priori as well as comparing against results of the Class B algorithms.

The constraining equations for the two autocalibration methods presented in this work are non-linear and based on the fundamental matrix. In what follows, we will show that it is possible to reformulate the process of autocalibration into the minimization of a cost function of the calibration parameters. While this type of reformulation has been achieved for class A algorithms and is clearly evident in Class C algorithms, this is not the case for class B algorithms. For example, in [17] the basis of the class B autocalibration algorithm is the modulus constraint. The modulus constraint is a non-linear relationship between the camera calibration parameters and the projective camera matrices that makes autocalibration possible [80]. The application of the modulus constraint produces a set of X polynomial equations for every pair of images, and a system of polynomial equations for the entire image sequence. Given an M image sequence, we have X^{M-1} equations in the system. The solution of such a polynomial system is very difficult to compute. One possibility is to find all the permutations of exact solutions in closed form and then to combine the results [79]. This is rather cumbersome. Another way to solve such a polynomial system is to use a continuation method [89]. Unfortunately, continuation methods only work well for a small number of equations, and are not suitable for the large polynomial systems generated by long image sequences. By contrast, the methods presented in this work are computationally efficient (with a known upper bound on the

number of times the cost function will be executed) even for large image sequences. Furthermore, the accuracy of these algorithms improves as the image sequence lengths increase.

In this work, we examine two class A autocalibration algorithms based on the fundamental matrices; one based on Kruppa's equation [75, 77, 79], and the second based on the idea of finding the calibration matrix which optimally converts a fundamental matrix to an essential matrix [78]. In both cases the problem can be formulated as the minimization of a cost function of the calibration parameters, which will be described in detail in Sections 3 and 4. The correct camera calibration is the global minimum of this cost function over the space of possible camera parameters. In the past, claims have been made that such minimization approaches to autocalibration are sensitive to the initial starting point of the gradient descent algorithm [76, 90]. However, when computing only one parameter, the starting point is irrelevant because we can accurately solve the associated 1D optimization problem using standard numerical approaches [91]. When there is more than one parameter, such as focal length and aspect ratio, we use a simple stochastic approach [92] from the field of evolutionary computing to overcome this problem. We show experimentally that for this type of cost function the stochastic method reliably finds the global minimum. As well, a number of experiments are performed on image sequences with known camera calibration. We compare the results of our method against Class B results on some of the same image sequences, and provide evidence that shows that the stochastic approach achieves results that are comparable.

Our first Class A algorithm relies on the fact that the fundamental matrix can also be decomposed into terms of the essential matrix and the camera calibration matrices as described by (5.1). Our second algorithm relies on the existence of the projection of the absolute conic within an image pair.

5.2 Autocalibration via Equal Eigenvalues

5.2.1 Single Image Pairs

The essential matrix can be considered as the calibrated version of the fundamental matrix. Given the camera calibration matrix K and the fundamental matrix F, then the essential matrix E is related by the following equation:

$$\mathbf{E} = \mathbf{K}^{\mathsf{T}}\,\mathbf{F}\mathbf{K} \tag{5.1}$$

Since F is a 3x3 matrix of rank two with the condition that there are exactly two non-zero eigenvalues, E is also of rank two. The essential matrix (E) however, has an added constraint that the two non-zero eigenvalues must be equal [93]. It is this constraint that is used to create the autocalibration algorithm [78]. The goal is to find the calibration matrix K that makes the two eigenvalues of E equal, or as close to equal as possible. Given two non-zero eigenvalues of E, σ_1 and σ_2 where $\sigma_1 > \sigma_2$, in the ideal situation ($\sigma_1 - \sigma_2$) should be zero. Consider the difference $(\sigma_1 - \sigma_2) / \sigma_1$, which can be rewritten as:

$$1-(\sigma_2/\sigma_1) \tag{5.2}$$

If the eigenvalues of E are equal, (5.2) computes to zero; as they differ, equation (9) approaches one. Clearly, (5.2) becomes the cost function to be minimized.

5.2.2 Multiple Image Pairs

Since we are dealing with a sequence of M images, we can have at most M-1 adjacent image pairs. Since a fundamental matrix is computed between each adjacent image pair we therefore have M-1 different fundamental matrices F_i (i=1..M-1). Based on our assumption that the intrinsic parameters of the camera do not vary, our goal is to find K by minimizing the cumulative values of (5.2) for all the fundamental matrices (F_i) in the sequence. Assume F_i is the fundamental matrix relating image i_k and i_{k+1}. To autocalibrate over the M image sequence, we must find the K that minimizes:

$$\sum_{i=1}^{M-1} \omega i_i (1-\sigma_2/\sigma_1) \tag{5.3}$$

Where ϖ_i is a weighting factor, between zero and one, which defines the confidence we have in the computed fundamental matrix F_i. The weights ω_i are set in proportion to the

number of matching 2D feature points that support a given fundamental matrix. The larger the number of 2D points that support the epipolar geometry characterized by F, the more confidence we have in that fundamental matrix, and therefore the smaller the weight (remember we are minimizing). Each weight ω_i is normalized to a range from zero to one.

5.3 Autocalibration via Kruppa's Equations

In a similar manner, we can convert Kruppa's equations into a cost function that can be used in either single or multiple image pairs.

5.3.1 Single Image Pairs

Another way to perform autocalibration from the fundamental matrix is to use Kruppa's equations [86, 93]. To understand these equations we must first define the absolute conic. In Euclidean space the absolute conic lies on the plane at infinity, and has the equation:

$$x^2 + y^2 + z^2 = 0 . \tag{5.4}$$

The absolute conic contains only complex points that satisfy the equation $M^TM = 0$. If we consider a standard camera projection matrix

$$P = K[R|-Rt]. \tag{5.5}$$

Where R is the rotational component of the motion of between camera positions and t is the translational component of the camera motion, then a 3D point x on the absolute conic projects to a 2D point:

$$m = P(M) = KRM. \tag{5.6}$$

Where

$$M = R^TK^{-1} m, \tag{5.7}$$

and since $M^TM = 0$, this implies:

$$m^TK^{-1}RR^T K^{-1}m = m^TK^{-T}K^{-1} m = 0 . \tag{5.8}$$

This clearly shows that any 2D point *m* is on the image of the absolute conic if and only if it lies on the conic represented by the matrix

$$K^{-T}K^{-1} \tag{5.9}$$

From projective geometry,

$$KK^{T} \tag{5.10}$$

is the dual absolute conic, and is labeled as C. If we can find C, then we can directly compute the camera parameters K by Cholesky factorization [94].

Kruppa's equations relate the fundamental matrix to the terms of the dual absolute conic. The first form of these equations required the computation of not just the fundamental matrix, but also of the two camera epipoles, which are known to be unstable [93]. Recently, a new way of relating the fundamental matrix and the dual absolute conic was described which does not require the computation of the camera epipoles [75]. Consider the singular value decomposition of a fundamental matrix F to be UDV^{T}. We let the column vectors of U and V be u_1, u_2, u_3 and v_1, v_2, v_3 respectively. This gives the new form of Kruppa's equation as:

$$\frac{v_2^T C v_2}{r^2 u_1^T C u_1} = \frac{-v_2^T 2C v_1}{sr u_1^T C u_1} = \frac{v_1^T C v_1}{s^2 u_2^T C u_2} \tag{5.11}$$

To autocalibrate we must find the C which makes these three ratios equal, or in the case of estimation, as close to equal as possible. We let $factor_1$ be equal to:

$$\frac{v_2^T C v_2}{r^2 u_1^T C u_1} - \frac{-v_2^T 2C v_1}{sr u_1^T C u_1} \tag{5.12}$$

And we define $factor_2$ and $factor_3$ similarly as the other two possible permutations of the system of ratios. Autocalibration can then be achieved by finding the C (KK^{T}) that minimizes the sum of the factors squared.

5.3.2 Multiple Image Pairs

Given the same M-1 fundamental matrices defined in the previous section then autocalibration with the Kruppa method over M images requires the minimization of:

$$\sum_{i=1}^{N-1} \omega_i (factor_1^2 + factor_2^2 + factor_3^2) \qquad (5.13)$$

Again, ϖ_I is a weight factor, between zero and one, which is the confidence in the computed fundamental matrix F_i as described in the previous section.

5.4 The Evolutionary Approach

Since the two autocalibration methods based on the fundamental matrix have an associated cost function we can use a gradient descent algorithm to find the solution. The caveat here is that there are often many local minima in the cost function, so the solution that is found depends on the starting point. However, we note that the calibration parameters can all be bounded; i.e. the center of projection rarely varies from the image center, the aspect ratio is generally one and the skew is almost always 90 degrees. Thus we are attempting to find the global minimum for a set of real-valued, bounded optimization parameters. This problem has been dealt with in the field of evolutionary computing.

There are many possible evolutionary approaches, but they are not all equally applicable to every problem. We use the ideas around Genetic Algorithms (GAs) [95]. The idea behind GAs is to simulate evolution by defining each solution as a chromosome, and then defining the appropriate crossover and mutation operators. While GAs are a very powerful framework, they must be adapted and tuned specifically for each application. In our application of function minimization the process of simulated annealing has also been successful [91]. The idea behind simulated annealing is to perform function optimization by simulating the process of annealing crystals; essentially by slowly lowering the temperature. The issue we face is: Which evolutionary approach is best? We define this problem to mean the simplest and most effective algorithm that arrives at the correct answer.

As the camera calibration problem is being recast as a parameter optimization problem for a set of real-valued, bounded optimization parameters, we use the dynamic hill climbing technique that combines the strengths of genetic algorithms and hill climb-

ing techniques that was specifically designed for this type of problem. Dynamic hill climbing (DHC) can be considered a hybrid evolutionary algorithm because the algorithm makes use of concepts such fitness, population expansion and mutation, but utilizes a hill climbing technique for determining local extrema. Also, by using a mutating coordinate frame combined with local extrema exploitation DHC has been empirically shown to outperform classical genetic algorithms, simulated annealing and typical hill climbers when optimizing parameters of the DeJong [96] test suite [92]. DHC optimization results on the DeJong test suite were independently confirmed in [97] and subsequently used in range image registration. The compared methods included genetic algorithms, simulated annealing and the DHC algorithm. Experimental results showed that the DHC algorithm was the most successful evolutionary approach for this type of bounded, real-valued function optimization. For the above reasons, we chose DHC and we describe the dynamic hill-climbing algorithm in detail next.

5.4.1 Dynamic Hill Climbing

The workhorse behind the DHC algorithm is simple, yet very efficient hill climbing algorithm and the use of population expansion via mutation to cover the search space. The process begins by selecting an individual randomly from the population (search space) and applying mutations to the single individual, expanding the population. The parent and all the offspring (mutations) are considered for the next generation, with the fittest individual from the family surviving. At each generation the age of individual is increased, however when the offspring are determined to be the fittest and selected for survival, they inherit the age of the parent i.e. the generational age. The mutations are performed by scalar adjustment to each of the coordinates in each direction. This means we perform 2N mutations in an N dimensional search space, keeping within any bounds that may limit the search space.

As the age of the population increases, the magnitude of the mutations proportionately decreases allowing convergence toward the local extrema, and a more thorough exploration near the local extrema as the population ages. While a variety of heuristics may be used to determine the magnitude of the scalar adjustment, we use a logarithmic halving of the bounded dimensions of the search space. This results in an upper bound of

O(log D) generations where D is the largest range within the search parameters. Furthermore, in an N dimensional search space there are N generations considered as the mutations adjust only a single parameter at a time. Finally, because each generation will perform the fitness evaluation 2N times, we have an upper bound of $2N^2\log(D)$ function evaluations and an upper bound of $O(N^2 logD)$ fitness function evaluations. Within the scope of camera calibration, we have an upper bound of the search space being 5 dimensional and a reasonable practical range for the parameter space, limiting D allowing us to determine a concrete upper bound on the time complexity for camera calibration.

5.4.2 Mutating coordinate frames

A static coordinate frame results in premature cessation at a local extrema (the foothill problem) because the hill climber cannot move in the direction necessary to reach the true extrema. For example, if a hill climber can move in only 4 directions, say the major compass directions, when a true extrema can be reached by moving in a northwest direction the classical hill climber will fail. DHC addresses this issue by allowing a mutating (dynamic) coordinate system. DHC keeps a historical record of previous movements and constructs a new basis via a Gram Schmidt orthogonalization of the last two positions. By doing this, DHC is able to adjust for directional changes within the structure of the search space, which avoids the foothill problem in certain cases.

5.4.3 Exploiting local optima

Dynamic hill climbing also tries to avoid early convergence to a local extrema by ensuring that diversity of the population is considered directly, and independently of the fitness function. Because the local hill climber has a mutation size that decreases with age, the local area is searched more thoroughly to help ensure that there is no other local extrema with better fitness. Once a local extrema is found, the individual is moved to a separate pool of static individuals that have found local extrema. When the search system stalls, DHC will examine the pool of static individuals who have achieved a local extrema and select a new population that is as different as possible from the static pool.

To facilitate this, DHC examines the Hamming distance (The number of differing bits) between the two individuals and tries to maximize the distance. We note here that it is possible that this strategy is not without its own problems, the following example illustrates this: Suppose a local extrema exists at 127, bit set 11111110, the maximum hamming distance results in bit set 00000001, or 128, which is not sufficiently far from 127. However, it should be noted that that a sufficiently large population reduces the probability of getting stuck when using this strategy of exploiting the local optima.

5.4.4 Coverage of Search Space

The basic idea in the DHC approach is to repeatedly perform gradient descent in the search space but to start the gradient descent in an area of the search space that is as far removed as possible from previous solutions. We call this principle of operation Statistically Distributed Randomized Starting (SDRS).

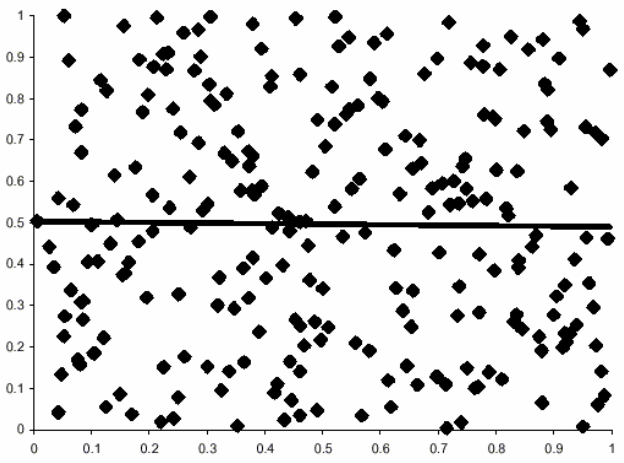


FIGURE 5.1: SCATTER PLOT OF 2D SEARCH SPACE GENERATED BY SDRS.
250 points with a trend line indicating an even disbursement of start points.

The effect is to cover the search space very thoroughly, and at the same time avoiding areas that have been previously explored and therefore avoiding the local minimum. This covers the search space very effectively, as is shown in Figure 2. In this Figure we show the start points of the gradient descent in a 2D SRDS process. It is clear from the distribution that the search space is uniformly explored.

SRDS covers the search space as completely as possible with a user specified number of starting points. Essentially SRDS is a simplified variation of dynamic hill climbing's exploitation of local optima. The only operating parameter is the number of repeated gradient descents to try, and this is manually set to be approximately one hundred. It is important to note that the range of the calibration parameters, focal length and aspect ration is bounded. In practice, the focal length is in the range of 1 to 10,000 pixels, and the aspect ratio is in the range of .5 to 2.0. Under these conditions and operating parameters the DHC algorithm has had good practical success.

The pseudo-code for SDRS is below:

```
SDRS()

For each parameter in the search space
    Find the largest region that has     not had a start point
    Compute a random point X in this
      region
    Set point X to the start point
      for this parameter
Endfor

Return N-dimensional startPoint for the next gradient descent (DHC)
```

5.4.5 Autocalibration Algorithm

The algorithm ESTIMATE_K returns the calibration parameters in the matrix K that produced the minimum value from the cost function. It is based on the SRDS and the DHC algorithms described previously. As we have shown in the previous sections, the actual evaluation of the cost function for the two different autocalibration methods is very efficient and the upper bound on the number of calls to these functions is also known to be $O(N^2 \log(D))$. The equal eigenvalues approach requires only the computation of the eigenvalues of a three by three matrix, and for the Kruppa approach the computation of three ratios based on the SVD of a 3x3 matrix. Furthermore, precomputing the SVD and storing them in a lookup table for use by the algorithm can further optimize the process and reduce the time required to execute the cost function. A single gradient descent of

the cost function uses the Powell optimization algorithm, which is in turn based on repeated applications of the one dimensional Brent method [91].

As we know the upper bound on the number of times the cost functions are called, we have an upper bound on the entire process of $O(N^2 logD)$, which is the upper bound for the DHC algorithm. The remainder of the autocalibration algorithm is simply the addition of constants affecting the computation time which are equal to the time required to execute 1 instance of the cost function. To be precise: Given an image sequence of M images, and computing N intrinsic parameters, bounded by a maximum range of D, the running time on the autocalibration will be no more than $O(MN^2 log(D))$ computations of the cost function. As we can see this is linear with respect to the number of images, as opposed to the exponential number of equations generated using the modulus constraint based methods.

The basic pseudo-code for estimating K:

```
ESTIMATE_K()
For n times
    StartPoint = SRDS()
    Perform the DHC gradient descent
      from StartPoint.
    IF Cost function (Equal Eig.
      OR Kruppa) is minimal
        Save this K.
    ELSE
        Discard this K
    Endfor

Return K
```

5.5 Degeneracy

The method presented makes use of all the computed inter frame geometries; however no consideration is given for incorrectly computed fundamental matrixes. An incorrect fundamental matrix can occur and is known as a degeneracy case. It is commonly known that there are degenerate situations where many epipolar geometries will support the same feature match set [98].

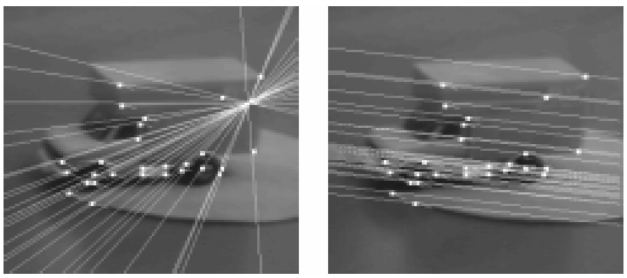


FIGURE 5.2: DEGENERATE EPIPOLAR GEOMERTY
Two epipolar geometries that support a feature match set, yet only one can be correct. (From [98])

As shown in Figure 5.2, we have 27 corresponding points and two computed epipolar geometries that support them. Clearly there can be only one truly correct geometry; however, it simply takes a single outlier to potentially produce an incorrect geometry. Clearly an incorrect fundamental matrix will result in an incorrect self-calibration when using only the one incorrect fundamental matrix.

The potential for computation of a single degenerate fundamental matrix from a sequence of images when using a RANSAC method is unavoidable and thus all computed geometries from an image sequence are to be considered. By simply using the fundamental matrix with the highest support, we will achieve incorrect results when that computed geometry is degenerate. By using all of the computed fundamental matrices, we have some knowledge on the effect each fundamental matrix has on the cost function. If we assume for demonstrations sake, that we have equal confidence in each and every fundamental matrix that has been computed for an M+1 image sequence. A single degenerate geometry will weigh in at 1/M and therefore only affect the computation proportionally to the number of images in the sequence.

5.5.1 Handling Degeneracy

While methods exist that attempt to detect degenerate configurations [98], we have chosen to use the number of supporting matches for each fundamental matrix as a measure of confidence. This metric, while not theoretically as reliable as a method that detects degeneracy, is suitable because the automated methods for computing the fundamental matrix [99] provide a relatively large number of matches with the associated fundamental matrix. Our experiments are performed under the assumption that the number

of feature matches used to compute the fundamental matrix reduces the likelihood of computing a degenerate geometry. We rely on the effectiveness of the software presented in [100] to produce many feature matches and compute fundamental matrices with sufficient support that the probability of outlier caused degeneracy is greatly reduced, yet any reliable computation of the fundamental matrix will have the same result. Therefore, we use magnitude of the support feature set that was used to compute the geometry as a measure of our confidence.

Degeneracy can also be effectively handled in other ways and we outline a couple of methods next. The first obvious solution is to use the PLUNDER algorithm (Pick Least UNDEgenerate Randomly) outlined by Torr in [98], however it is more complicated to implement than other solutions. The benefit of handling degeneracy this way is that we can be sure that all fundamental matrices we are using are not degenerate. Another alternative is to prune fundamental matrices that produce calibrations parameters that are not consistent with the entire set. Effectively we perform a single image pair calibration for each fundamental matrices in the sequence and perform a statistical analysis of the individual results. We can now prune any fundamental matrix whose individual calibration results are outside an acceptable level of error. Using covariance analysis or Frobenius norm will provide reasonable results.

5.6 Experiments

There is no practical reason to autocalibrate all five intrinsic parameters [88], however, by assuming the principle point and the skew are fixed, results are encouraging. This problem is not unique to our method, and occurs in Class B algorithms as well [81]. In [81], the principle point could not be computed accurately using the Class B algorithm and for this reason it was also assumed to be fixed.

For many autocalibration algorithms the evaluation of performance consists of a simple visual inspection of the resulting 3D reconstruction. This is not an adequate metric because it has been shown that the quality of the final reconstruction is visually acceptable for a wide variety of calibration parameters [88]. In order to test the capabilities of

the presented evolutionary method, we used test data for which the ground truth was known; i.e. the intrinsic parameters are already known apriori. Some of these data sets are the same ones used in the literature, in particular those for the class B algorithms. The conclusions are that the results of the Class A algorithms using the evolutionary approach is comparable to that of the Class B algorithms yet the simplicity and efficiency of the evolutionary method is significant. The experimental results also give an indication of what the autocalibration errors are for a typical image sequence. We performed these experiments a number of times, to make sure that the results of the SRDS algorithm are repeatable and unbiased.

The first set of experiments described in Table 1 show how the autocalibration process works when we are calibrating only the focal length. Table 1 shows the results for a number of different test sequences that have been processed in previous autocalibration papers [17, 77, 79, 101]. In particular, the Castle sequence [79] is used as a test case for comparison with the class B approach that requires a projective reconstruction. We see that our autocalibration results are comparable to those of other class B self-calibration algorithms.

In Table 1 we list our autocalibration results compared to the previously published results in the literature, which we assume to be correct. In the last example from [101] shown in Table 1, the error with the Kruppa autocalibration is quite large. A possible explanation is that the motion is close to being a pure translation, which is known to be a degenerate motion for the Kruppa algorithm [86, 87]. It is also a good indicator of how the Equal Eigenvalues method performs well in spite of these degenerate motions. In these experiments we take the image sequences as input and compute the matching feature points automatically, using the software described in [100]. In other words we are not given matching 2D feature points, but simply a set of images. Therefore the closeness of our results to those published in the literature is significant because we are actually using different software to compute the fundamental matrices. We also are unable to verify independently that the published ground truth focal lengths are correct, it is possible that the stated focal lengths have some level of error in them as well.

In the next set of experiments outlined in Table 2, the 2D feature points were selected by hand as part of a photogrammetric model building process. From these manually selected correspondences we compute the fundamental matrix between all image pairs in the sequence. In this experiment we know the intrinsic parameters of the camera a-priori from the project parameters of the photogrammetric package [41]. We therefore assume that all the intrinsic parameters are set apriori, except for the focal length which we autocalibrate. Table 2 shows the autocalibrated focal length in millimeters versus the true focal length, along with the percentage error for both autocalibration methods. Since we have the associated 3D reconstructions for the corresponding 2D features, we can use more sophisticated performance measures, namely reprojection error.

For a given autocalibrated focal length we compute the reprojection error for all the corresponding feature points. The reprojection errors are the pixel differences between the projection of the 3D feature points into 2D and the original corresponding 2D features. We compute the median of the reprojection errors using the correct focal length, the focal length found by the eigenvalue method, and the focal length found by Kruppa's method. The median of the reprojection errors is a good indicator of the quality of the □pipolar□ction for a given focal length. We see that the median reprojection error increases for the autocalibrated focal lengths, but only slightly. This implies that the error in the autocalibrated focal lengths would not have a significant impact in terms of reconstruction quality; this independently verifies the work of Bougnoux [88].

In the next experiment we attempt to autocalibrate both aspect ratio and focal length using the two class A methods. We are again using as input a series of photogrammetric projects for which we know the 2D feature correspondences as well as the ground truth.

While the results as shown in Tables 3 and 4 are reasonable, the errors when autocalibrating two camera parameters are sometimes higher than autocalibrating just one parameter. The error again compounds when we attempt to auto calibrate all parameters. In

particular, the percentage error in the focal length increases slightly. One possible explanation is that the gradient descent algorithm is stuck in a local minimum. To verify this, the results shown in these two tables were computed by averaging over one hundred separate runs of the optimization algorithm. The variance as shown in Tables 3 and 4 for the autocalibrated aspect ratio and focal length is very small over these runs. This indicates that it is highly likely that the stochastic optimization algorithm is finding the correct global minimum.

Table 1: Results of autocalibration for focal length vs other algorithms. Focal length is in pixels. Correspondences are computed automatically.

Name	# of Images	Stated Focal	Computed focal length (Eq.Eigen)	% error vs. Stated	Computed focal length (Kruppa)	% error vs. Stated
Castle	27	1100	1156.50	5	1197.7	8
Valbone	9	682	605.5	11	685.71	0.5
Nekt	6	700	798.58	14	872.44	24.6
etluueshiba	5	837	857.25	2.4	1233.85	47.4

Table 2: Results of autocalibration for focal length for photogrammetric sequences. Focal length is in mm., and reprojection error is in pixels. Correspondences selected by hand.

Name	# of Images	True focal length	Eigen focal length	% error	Kruppa focal length	% error	Correct reproj.	Eigen reproj.	Kruppa Reproj.
Curve	4	6.97	4.71	32.4	7.49	1.13	7	2.23	1.44
Cylinder	3	28	26.35	5.9	31.70	13.21	0.96	2.07	2.60
Plant	6	24.20	22.55	6.8	24.39	0.78	0.80	1.49	1.04
Statue	7	5.11	3.67	28.2	5.29	3.5	3.93	9.61	1.95

Table 3: Results of autocalibration for focal length and aspect ratio for photogrammetric sequences. The equal eigenvalue method is used and focal length is in mm.

Name	True aspect	Eigen Aspect	Variance	% error	True Focal	Eigen focal	Variance	% error
Curve	1.0	1.08	0.003	8	6.97	3.46	0.062	50
Cylinder	1.0	0.98	0.002	2	28	26.72	0.52	4.5
Plant	1.0	0.98	0.012	2	24.2	22.96	0.39	5.1
Dam	0.81	0.972	0.0001	20	30.75	38.52	0.089	9.8

Table 4: Results of autocalibration for focal length and aspect ratio for photogrammetric sequences. The Kruppa autocalibration method is used.

Name	True aspect	Kruppa Aspect	Variance	% error	True Focal (mm)	Kruppa focal (mm)	Variance	% error
Curve	1.0	0.997	0.011	1.3	6.97	7.56	0.21	8.4
Cylinder	1.0	1.03	0.0001	3	28	32.91	0.0001	17.5
Plant	1.0	0.92	0.003	8	24.2	26.33	0.12	8.8
Dam	0.81	0.997	0.0001	19.75	30.75	38.43	0.0001	24.9

Table 5: Results for autocalibration of focal length for three sequences taken from the same uncalibrated camera.

Name	# of Images	Eigen focal	Kruppa Focal
Chapel	12	27.82	31.31
Climber	13	27.91	33.88
Workshop	8	26.19	38.09

The next set of experiments, shown in Tables 5,6 and 7, have as input image sequences that were taken with the same camera with invariant intrinsic parameters. There are image sequences that we have taken by hand, for which ground truth is known, or from various other modeling projects [102]. In these experiments we again compute the correspondences automatically using the software described in [100]. Test cases Chapel and Workshop are almost pure translation while the Climber sequence has a motion with significant translation and rotation. We autocalibrate only the focal lengths, which should be equal for all three sequences. The variance of the computed focal length for the eigenvalue method is 0.96 mm and for Kruppa approach is 3.42mm. It is not surprising that the autocalibration results differ, since certain motions are degenerate with regards to the Kruppa based autocalibration [86]. What these results clearly show is that for a given camera, and substantially different sequences, the evolutionary algorithms (especially the equal eigenvalues method) are convergent. Furthermore, longer sequences converge with a more accurate estimation of the intrinsic camera parameters.

The final set of experiments, shown in Tables 6 and 7, has as input image sequences that are used as test data for the ISPRS Working Group V/2 on Scene Modelling and Virtual Reality [102]. These images are used to test different model building software

packages, and the ground truth is known. In Tables 6 and 7, we again compute the correspondences automatically using the software described in [100], and autocalibrate only the focal length. We see, in Table 6 that the results are reasonable given that the true focal length is 1737 pixels in all cases, but that sometimes Kruppa's approach does not converge. The likely causes are sensitivity to motion degeneracy and the difficulty of convergence with a small number of images associated with the Kruppa method.

Table 6: Results for autocalibration of focal length for three sequences used by the ISPRS Working Group on Scene Modelling and Virtual Reality.

Name	# of Images	Eigen focal	Kruppa Focal
Indoor	5	1663	1815
Waterways	3	1759	fail
Building	2	1609	fail

Table 7: Results for autocalibration of focal length and comparison to ground truth.

Project	Focal length (in pixels)	Computed Focal length (in pixels)	# of Images	% Error
Amsterdam	1736.7	1866.7	4	7.48
Benches	1736.7	612	7	64.76
Chapel-l	2105	1640	2	22.0
Chapel-S	2105	1473	7	30.0
Corfu	2923.4	2995	7	0.02
Fitting	1684	1681	2	0.001
Florence	1897.3	1787	6	0.058
Light	2348.3	3647	4	55
Nikh	2348.3	2348	2	0.001
Oldbuild	1649.5	1588	7	0.037
Reg-1	2095	1609	7	23.1
Reg-2	611	747	27	22.2
Sphinx	1754	1764	16	0.0057

Table 7 presents a variety of experiments also from the ISPRS workgroup. In certain examples that error is very large, however the average error is only 17.25 percent with a standard deviation of 21.99. By removing the two grossly incorrect samples from the table the percent error and standard deviation drop in almost in half to 9.54 and 12.11 respectively.

In summary, Table 1 shows that the evolutionary approach is as good as the published results for Class B algorithms, particularly the castle sequence. However, the class B algorithms are not easily scalable from a computational point of view, and thus cannot handle long image sequences. The class A, fundamental matrix based, approaches are very efficient computationally because single evaluations of the cost functions do not take long and accuracy increases as the sequence length increases. The time taken for autocalibration is in the order of seconds for all the image sequences on a 400 MHz Pentium II processor. It seems that the equal eigenvalues method is superior to the Kruppa's method for degenerate motions and smaller sets of images. There are cases, however, where the Kruppa's method clearly outperforms the equal eigenvalues method. Further investigation is necessary to determine whether or not a heuristic can be developed to choose one algorithm over the other by pre-determining the camera motion using arbitrary intrinsic camera parameters in a first step and using this knowledge to select an appropriate Class A, B, or C algorithm that using an evolutionary approach.

5.7 Conclusions and Discussion

This work presents an algorithm for self-calibration that has four major advantages:

1) Simplicity (and ease of implementation)
2) Accuracy and Reliability
3) Scalability (handles very long sequences)
4) Speed of Execution (known upper bound)

In theory, the autocalibration methods that use fundamental matrices should not perform as well as those that use the camera projection matrices of a projective reconstruction [86, 87, 93]. However, we show that for non-degenerate motions both methods perform equally well when we are calibrating only the focal length, or the focal length and aspect ratio. The equal eigenvalues approach, combined with evolutionary methods is very simple and performs as well as any Class B method we compared it against. While it is theoretically equivalent to the Kruppa approach, it performs better numerically in situations where we are closer to degenerate motions, such as pure translation and seems to converge better for smaller sets of images. Experimentally we have shown that evolu-

tionary based autocalibration using class A algorithms produces similar results to their class B counterparts.

We have shown that in practice the Statistically Distributed Random Starting (SDRS) helps to find the global minimum of the cost function reliably. We have also shown that the error in the autocalibration of the focal length is usually in the range of to 15%. This is adequate for applications in which the final results are used for visualization purposes, such as model building but clearly not for applications that currently require exact depth information.

When dealing with long image sequences, class B algorithms will produce a set of polynomial equations for each image pair. This results in a large system of equations for the entire image sequence. Continuation methods can solve small systems of equations but are ill posed when the number of equations becomes large. The methods proposed in this work have advantages for long image sequences. The methods we have described are computationally efficient with a known upper bound that is better than any published class B method on long image sequences and produces comparable results. It is also the case that processing long image sequences is advantageous in that any error for an individual fundamental matrix (because of a degenerate motion for example) will have less of an impact on the final result. For example, an M image sequence has M-1 adjacent pairs and therefore M-1 representative fundamental matrices. As M becomes larger (i.e. then number images in the sequence increases) the individual error associated with a single image pairs has less effect. The accuracy of the estimation only increases with the size of the image sequence. As the sequence length tends to infinity, the error can be more closely associated to the error within the individual computation of the fundamental matrix. Another advantage of long image sequences is that the global optimum is better defined than when using short image sequences. In other words with long sequences the global optimum tends to be sharper and better defined making the results more stable.

Due to a lack of standardized data sets that can be used to effectively benchmark different autocalibration routines; the “look” of a resulting reconstruction is often used as

a benchmark, which is not appropriate for performance evaluation. For proper performance analysis of autocalibration algorithms it would be very useful to have a standardized set of images for which the ground truth is known. A start has been made in [26], but more needs to be done. At the very least, results using such test data should include the accuracy of the parameter values, consistency of results (similar to experiment 5), and an accuracy to image sequence length ratio benchmark.

Evolutionary based autocalibration with varying intrinsic parameters still remains an open problem; however it is conceivable to adapt the cost functions to allow for varying focal lengths between image pairs.

Chapter 6

Synchronizing Multiple Video Sequences

6.1 Introduction

There are many common applications of multiple video cameras today that range from video surveillance of large areas such as shopping centers, parking lots and campuses, to videography and filmmaking that utilize multiple video cameras when shooting individual scenes of a screenplay. However, in some situations such a photogrammetry and camera metrology, the use of multiple cameras is required when there are moving objects in the scene [1]. As different human operators may control these cameras, and only in certain situations is it feasible to use a professional camera synch slate, there is the fundamental problem of sequence synchronization that needs initial resolution.

Figure 6.1: Camera Sync Slate, a.k.a. clapper board[8]

Intuitively, the synchronization problem refers to the following: Given *k* different video sequences that overlap in time, identify one frame from each of the different sequences that refer to the same point in time. Such a set of frames is called a *synchronized cross camera subset*. More formally, for each video sequence *i*, let the *frame-time* function $T_i(f)$ map an integral frame number *f* of sequence *i* to a universal time, i.e.

$$T_i(f) : N \rightarrow R \tag{6.1}$$

The synchronization problem can now be expressed as finding a set of frames numbers, $f_1, f_2, \ldots, f_k$, one from each video sequence, such that the *synchronization equality* $T_1(f_1)$

[8] Picture from Filmtools, Burbank CA. http://www.filmtools.com/ Used with permission.

$= T_2(f_2) = \ldots = T_k(f_k)$ holds. Such a set of frames that exactly solves the synchronization equality is said to be in *perfect integral synchronization*.

However, due to possible minute variations in camera start times and variations in frame rates, perfect integral synchronization does not always exist. In such a case we search for a set of frames whose pair-wise difference with respect to the synchronization equality is minimized. We provide exact bounds on the value of this pair-wise difference in the next section. If we remove the restriction of integral frame numbers, the frame-time function maps frame values (integral and non-integral) to a point in time, i.e:

$$Ti(f) : R \rightarrow R \tag{6.2}$$

In this case, the frame-function maps a real frame number (sub-frame accurate) to an exact moment in time. In such a case, there will always exist a set of real frame numbers, $f_1, f_2, \ldots, f_k$, one from each video sequence such that synchronization equality $T_1(f_1) = F_2(f_2) = \ldots = T_k(f_k)$ holds.

In summary the synchronization problem is:

1. …referred to as the *full frame synchronization problem* when restricted to integral frame numbers, and seeks to minimize the pair-wise differences of the *synchronization equality*. i.e. $|T_i(f_i) - T_j(f_j)|$ is minimal for all sequence pairs i,j.
2. …referred to as the *exact synchronization problem* when unrestricted, and seeks to exactly solve the *synchronization equality*.

We fully explore the details of the functions, the equality and their use in both flavours of the synchronization problem in section 6.2.

6.1.1 Additional Background

Synchronization is often assumed [104], however, since the processing of large volumes of video data is becoming tractable, recent work has investigated the problem of synchronizing video sequences. In [105] the synchronization problem is constrained to having a large planar surface present. The method computes the homography that describes the transformation of the ground plane and looks for the frame pair with the most consensus of the moving objects. The method suffers under certain 3D motions such as similar objects moving in a line with constant velocity. In [106], the method is also constrained by a large ground plane being present, but further requires intrinsic camera pa-

rameters so that the 3D information about trajectories can be computed and subsequently corrected in conjunction with the epipolar geometry. Furthermore, the method assumes a homogenous camera system. In [107] a fixed set of extrinsic camera parameters, identical frame rates, and a static scene are required so that motion of the rig is identical on a frame to frame basis. This allows the algorithm to simply find the matching geometric changes between frames N and N+1 for camera 1, M and M+1 for camera 2, leaving the offset in frames being |M-N|. In [108], the authors also take advantage of the fact that objects moving on a planar surface produce a 3D trajectory contour that is identical from camera to camera. Upon finding the contour similarities, the frame synchronization is identified. Under repeating motion the contours will be identical, and synchronization will not be possible. In [109], the synchronization is based on viewing similar non planar 3D motion trajectories in time with applications to telelearning so that exact precision in synchronization is not fundamentally necessary. In [109, 110], the imposition of rank constraints on corresponding frame features is examined, rather than the epipolar geometry. In order to determine the synchronization a search is performed for frame pairs that minimize the rank constraint. However, in robust computations of the epipolar geometry, the rank constraint should be enforced.

Generally speaking these methods are restrictive because of the requirements of large planar surfaces being present or the requirement that the camera system be partially, if not fully, calibrated. Furthermore, as synchronization is simply a means to an end, they examine the full frame synchronization problem rather than the exact synchronization problem. In this work we examine the theoretical nature of the synchronization multiple video sequences and prove the maximum upper bound on the difference between full frame and exact synchronizations. We propose a novel method that handles a much larger set of input sequences and does not rely on any particular camera configuration or constraints on the objects. The method is performed solely in projective space and does not require trajectory correspondence to be solved apriori. The main constraint of our method is that there are at least three cameras that remain stationary throughout the video capture process; a very common situation in many multi-video applications. The motion of the moving objects is slightly constrained in that it cannot have a periodic characteris-

tic such as a pendulum, nor can the motion be directly along the optical axis of one of the cameras. Any camera count over three can be handled by our method on an overlapping basis.

In this chapter we examine the following: Section 6.2 formalizes the problem of synchronizing video sequences and introduces terminology. Section 6.3 outlines our proposed method that includes 1) Computing camera geometries, 2) Generating trajectories, 3) Using inflection points to grossly approximate the synchronization, and 4) using the computed geometries to compute the exact synchronization. Section 6.3 ends with an adaptation to handle errors in the computed geometries. In Section 6.4, we perform experiments with our proposed method and present results. In section 6.5, we examine some practical issues and finally we draw some conclusions.

6.2 Problem Formalization

We begin by formalizing the synchronization problem, and follow up by introducing terminology used in the description of the proposed solution.

6.2.1 The synchronization problem

We first examine some properties of the relationship between multiple video sequences and specify terminology. We let $F_i:R\rightarrow\{0,1\}$ be the *frame-capture function*. $F_i(x) = 1$ if at time x, a frame in video sequence i is being captured and $F_i(x) = 0$ otherwise. Close examination reveals that the frame-capture function is periodic in nature and therefore the model for video capture and synchronization we use is wave based, not linear as one might expect. The time between peaks in the function F_i is known as the *period* (in wave mechanics terminology) and what is commonly referred to as the *frame rate* (ρ), is actually the *frequency*. Recall that frequency and the period are inversely related. Figures 6.2 and 6.3 plot the function F_i. The peaks (value 1) occur when a frame is captured and the valleys occur (value 0) when frames are not being captured. Notice that in the case of multiple video sequences there exist what we call *primary synchronization points* that minimize the distance between the exact synchronization times and the full frame synchronization times.

Defintion: A *primary synchronization point* is a point in time that minimizes the difference between the exact synchronization function (6.2) and the full frame synchronization function (6.1) for all sequences. i.e. The frame numbers that satisfy the synchronization equality and minimize the difference given by (6.3).

Formally, the difference between full frame and exact synchronization is given by:

$$\left|Ti(fi) - Ti(\lfloor fi + 0.5 \rfloor)\right| \tag{6.3}$$

Any synchronization time that does not minimize the difference (6.3) is termed a *secondary synchronization point* i.e. any non primary synchronization point. As we see in Figure 6.2, given 3 video sequences of differing frame rates that are perfectly synchronized in time, clearly visible cycles of primary synchronization occur. For perfectly synchronized video sequences, these primary synchronization points correspond to the full frames that were taken at the exact same moment in time.

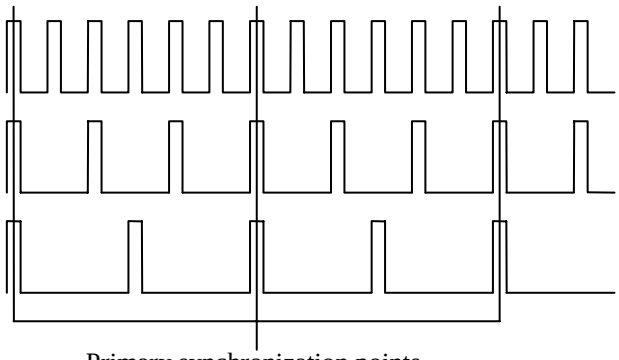


Figure 2: Perfect integral synchronized video sequences with varying frame rates showing primary synchronization points. .

Primary synchronization points occur at regular intervals that are a function of the frame rates of the individual sequences. The number of frames between these primary sync points for two sequences is a function of the frame rates given by:

$$frames(\rho_1, \rho_2) = \frac{\max\{\rho_1, \rho_2\}}{\min\{\rho_1, \rho_2\}} \tag{6.4}$$

The time between two primary synchronization points (λ) is determined by the maximum frame rate and the least common multiplier of (6.4) for all pairs of sequences.

$$\lambda = \rho_{\max} \bullet LCM\left\{frames(\rho_i, \rho_j) \mid \forall ij \text{ s.t. } 1 < i < j < N\right\} \tag{6.5}$$

We coin the term *primary synchronization period,* denoted by the symbol λ, to be the time between these events.

In practice however, we do not always have perfectly synchronized video sequences as shown in Figure 6.2. Instead, we have a slight synchronization offset. We can see in Figure 6.3, for sequences that are slightly out of sync, that the offset is minimal at the primary synchronization points. Furthermore, this offset has a maximum bound for any secondary synchronization point. We explore this bound next.

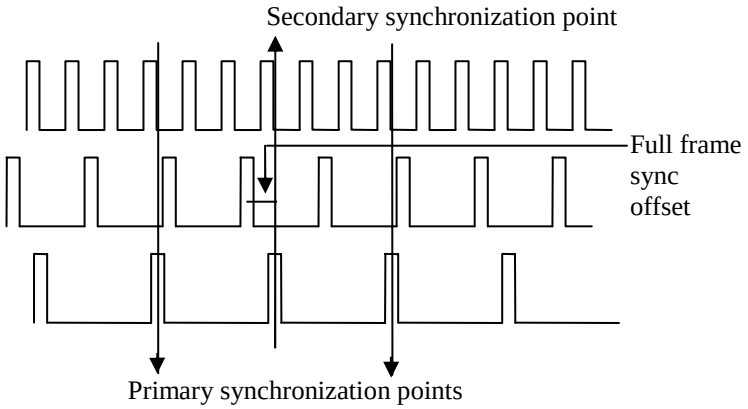


Figure 6.3: Imperfectly synchronized video sequences with varying frame rates showing primary and secondary synchronization points.

Given two imperfectly synchronized video sequences, the maximum full frame sync offset is half the maximum difference between two frame captures of the higher frame rate sample. As we can see in Figure 3 for any pair of sequences, a frame in the slower frame rate sample straddles two frames of the higher frame rate sample, and thus full frame synchronization will be with the closest frame, in time, of the higher rate sample. For two video sequences, the quality of full frame synchronization is bounded by:

$$offset(V_1, V_2) = \frac{\min\left\{\frac{1}{\rho_1}, \frac{1}{\rho_2}\right\}}{2} \tag{6.6}$$

For N video sequences, the error is bounded to a maximum error defined by (6) for all camera pairs and is characterized by:

$$\Delta = \max\{offset(V_i, Vj)\} \ \forall ij \text{ s.t. } 1 < i < j < N \tag{6.7}$$

It turns out that the maximum error will always be between the slowest and the 2nd slowest video frame rates, and thus for N cameras, Δ can be easily determined using (6.6) and the two slowest frame rates.

Lemma 2.1 *For N video sequences, the maximum full frame offset is bound by half the 2nd slowest camera period.*

Proof: Let ρ_1, ρ_2 and ρ_3 be the three slowest frame rates from N sequences such that: $\rho_N < \ldots < \rho_3 < \rho_2 < \rho_1$. For all sequence pairs, the offsets defined by (6.5) are $\rho_2/2$, $\rho_3/2$, and $\rho_3/2$ respectively. Since ρ_2 is greater than ρ_3, $\rho_2/2$ is greater than $\rho_3/2$. As ρ_3, ρ_2 and ρ_1 are the three slowest rates, any other frame rate ρ_i (i>3) from the N sequence is less than ρ_3 resulting in an application of (6.5) resulting in $\rho_i/2$ which is less than our largest value $\rho_2/2$. Therefore, the error is bounded by $\rho_2/2$, half of the 2nd slowest frame rate. □

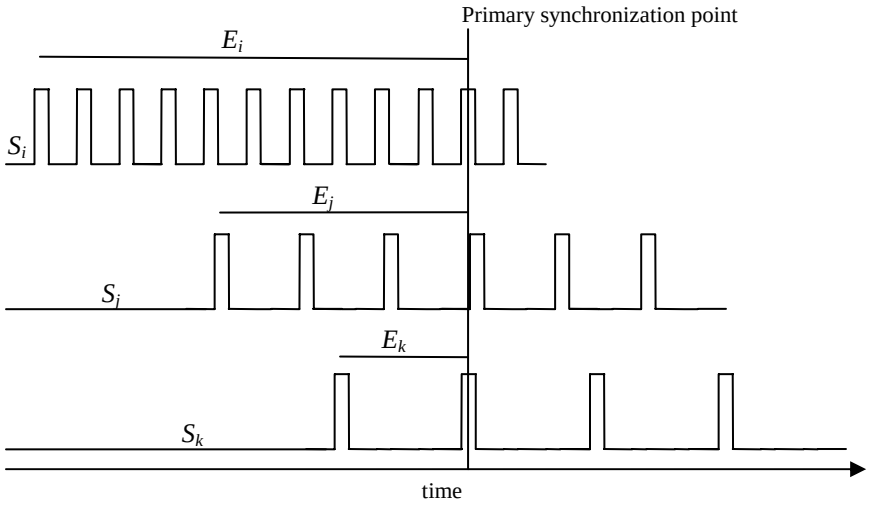


Figure 4: Video sequences with respect to a universal time line.

Given two frames from a single video sequence i, the amount of time that elapses between frame f_1 and f_2 is $(f_2-f_1)*\rho_i$. We let E_i represent the amount of time that has elapsed between the first frame and the nth frame (denoted n_i) in sequence i. Specifically, the n^{th} frame (n_i) in sequence i will be taken at elapsed time E_i and is given by the following equation:

$$E_i = n_i \cdot \rho_i \tag{6.8}$$

Because the sequence start time is the beginning of the sequence, we have a simple linear relationship between the frame rate and the frame number. However, since we want to synchronize video cameras that were not necessarily started at the same point in time, it is

necessary to determine the elapsed time within the context of a universal timeline, and not simply within the time line of the single sequence itself. We can now specify the exact nature of the function described in (6.1) and (6.2) by by:

$$Ti(n_i) = E_i + S_i \tag{6.9}$$

where the start time of the sequence *i*, is at some offset S_i from the universal start time. This offset in the universal time line represents a *phase shift* in wave mechanics terminology. As we see in Figure 4, three cameras started at different points in time have different phase shifts with respect to the universal time line. We see that there are phase shifts S_i, S_j, and S_k that correspond to the differences in time for which the cameras started capturing video sequences.

Given multiple video sequences, the synchronization consists of the frame numbers that were taken at the same instant in universal time, within the known bounded error Δ given in (6.7). For 3 video sequences *(i, j, k*), the universal time line obeys the synchronization equality:

$$Ti(n_i) = E_i + S_i = E_j + S_j = E_k + S_k \tag{6.10}$$

The problem of synchronization is now, in fact, two fold. 1) finding the inter-sequence times (*Ei*) where a synchronization point occurs and 2) solving for the universal time phase shifts (S_i). In practice, we can impose the constraint that S_1 be set to universal time 0 and base our phase shift values on a time frame dictated by camera start events. We can determine the camera start order by examining the elapsed sequence times in the order of highest to lowest.

The goal of synchronization is now to solve for the synchronization equality (6.10). This can be done on an integral frame basis, knowing that we can only be accurate to within the time frame given by (6.7), or it can be solved exactly by allowing sub frame accuracy determination in equation (6.10). If we choose to only support integral frame numbers, equation (6.10) has constraints on the determination of the inter-sequence times that account for the maximum error Δ.

$$\begin{aligned} \left|(E_i + S_i) - (E_j + S_j)\right| &\le \Delta \\ \left|(E_j + S_j) - (E_k + S_k)\right| &\le \Delta \\ \left|(E_i + Si) - (E_k + S_k)\right| &\le \Delta \end{aligned} \tag{6.11}$$

Additional cameras are a simple extension of the equality from (6.10) and the constraints from (6.11). Primary synchronization points minimize the constraints given by (6.11), and in practice should be sought.

The E's are solved by finding a primary synchronization point where each frame was taken at the same moment in time of the same 3D scene, and the S's by setting S_1 to be zero, therefore becoming the universal start time, and using algebraic manipulation to solve for the remaining.

2.1 Terminology

We continue by specifying terminology that defines the core concepts behind the proposed solution. A *camera sequence* (CS) is the linear sequence of frames from a single video camera; like a single reel of film. A *cross camera subset* (CCS) is a set of N images, where each image in the subset comes uniquely from one of the N cameras. A *synchronized CCS* is a cross camera subset where each frame of the set is full frame synchronized as outlined in equation (6.1).

A cross camera subset is not necessarily aligned in time, we denote a CCS to be simply a selection of N frames, one from each of N camera sequences. The problem of camera synchronization is that of determining the exact the same moment in time for each of the video sequences, i.e. finding a synchronized CCS.

We further sub-classify cross camera sets into dynamic-CCS and static-CCS. As their names elude, a static-CCS is comprised of those images that have the same static content (or in practice, a majority of static content). There are multiple static-CCS candidates among a set of video camera sequences. The term static-CCS is not to say that there is no dynamic motion within the frames, but rather we are interested only in the static content of each frame so that we can compute the camera geometries. A dynamic-CCS is the set of images in which we are utilizing the dynamic aspects of the cross camera set. Again

this is not to say that all pixels within a set of candidate frames are moving, but rather they contain the same moving objects.

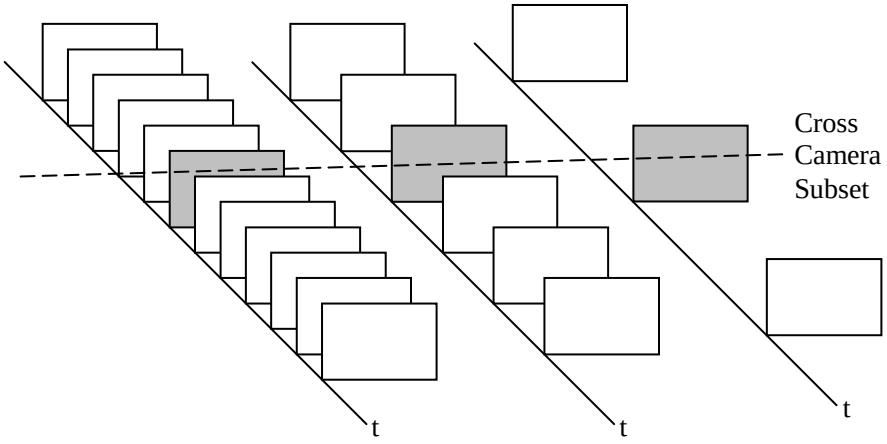


Figure 6.5: 3 camera sequences in a three video camera setup with varying frame rates, with a synchronized cross camera set in gray.

6.3 Recovery of the synchronization

Because we are utilizing multiple non-moving video cameras, we can use the fundamental matrices [9] and the trifocal tensor [10] of the three views to determine the synchronization offsets. There is only one instant in time where all moving and non-moving features will have perfect consensus on the camera geometries, and this is when the moving objects are captured at the same instant in time. When dynamic-CCS objects concur with the geometry computed with static-CCS, the frames that comprise the dynamic-CSS are full frame synchronized. Moreover, the best geometric support will come from a primary synchronization point. It is also important to note that more than one dynamic-CSS will support the geometry computed from the static-CSS; those being any dynamic-CSS that contain synchronized frames.

Clearly a pure brute force method (using all frame permutations) of finding the dynamic-CCS that supports our computed camera geometry is not an option since the combinations are exponential to the number of cameras. This would require M^N combinations to be examined, where N is the number of cameras and M is the frame count. One way to reduce the cost of the brute force method is to align groups of 3 adjacent cameras. With a maximum synchronization offset of just 30 frames (±15), a three camera

system will yield 27,000 combinations to be tested. This still remains computationally intense. In general, the number of combinations required to perform such a computation for an N camera sequence with a maximum offset of *max_offset* frames, would be:

$$(N-2)\cdot max\_offset^{3} \tag{6.12}$$

We alleviate the need to perform these brute force computations (even smarter brute force) by creating a virtual image that embeds the dynamic object trajectories in 2D. We utilize these virtual trajectory images to quickly determine the synchronization. The core idea we utilize is that the camera geometry and the object trajectories will concur, allowing us to quickly compute the frames with maximal geometric consensus and therefore implicitly generate the synchronization offsets. We use a basic 4 step system: 1) compute the camera geometry from a static-CCS, 2) generate trajectory images for each sequence, 3) Narrow down trajectory images via inflection points, 4) Refine the selection via consensus to single frame accuracy and via pure geometric support to sub frame accuracy.

6.3.1 Computing Camera Geometry from the Static-CCS

Because the cameras are static, and the features considered in a static-CCS are also stationary and we can select any frames as candidate frames so long as they minimize the effect of the moving objects. There are a variety of ways to achieve this, from a brute force examination of the frame data using some difference metric, to a user selected set of frames. Selecting static features that do not change from frame to frame, i.e. background subtraction, or utilizing optical flow methods to remove pixels that are not static, simply adds computational overhead that is practically not necessary.

Selecting frames that are relatively close to the synchronized frames should be avoided in this step to prevent outliers from being included, resulting in a degenerate computation of the camera geometry. However, due to the large ratio of frames to cameras, and this will be true in most practical cases, we can simply sample frames from each camera sequence so that they are well distanced in time.

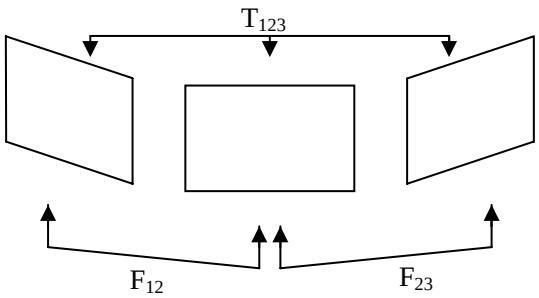


FIGURE 6.6: REQUIRED GEOMETRIES FOR A 3 CAMERA SYNCHRONIZATION

Once the static-CCS has been selected, it is used to first compute information about the camera geometry. In Figure 6, we see the required geometries to compute the synchronization of three video sequences. The fundamental matrices F_{12}, and F_{23}, and the trilinear tensor, T_{123}, are required from a 3 camera system and can be computed robustly using techniques outlined in [9][10], furthermore, one can simply use the tensor alone since F_{12} and F_{23} can be derived from T_{123}. We use the software presented in [11] for our experiments.

6.3.2 Generating the Trajectory Images

We utilize a feature tracking mechanism to generate the trajectory images. As we track features over time, we associate the current frame number to the position within the trajectory image. Feature tracking is performed on the luminance channel (grey map) for the video frames. The luminance channel is computed as follows:

$$\text{Luminance} = \text{Red}*0.299 + \text{Green}*0.587 + \text{Blue}*0.114 \quad (6.13)$$

The feature tracker we use is based on the work of Lucas and Kanade in [49]. This work was further developed by Tomasi and Kanade in [50] of which Shi and Tomasi provide a complete description in [51].

Briefly, features are located by examining the minimum eigenvalue of a 2x2 image gradient matrix. The features are tracked using a Newton-Raphson method of minimizing the difference between the two windows around the feature points. We continue by presenting a very brief outline of the work by Tomasi et al [49,50,51].

Given a point p in an image I, and its corresponding point *q* in an image J, the displacement vector δ between *p* and *q* is best described using an affine motion field:

$$\delta = Dp + t \tag{6.14}$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \tag{6.15}$$

is a deformation matrix and t is the translation vector of the centre point of the tracked feature window. The translation vector t is measured with respect to the feature in question. Tracking feature *p* to feature *q* is simply the problem of determining the six parameters that comprise the deformation matrix D and the translation vector t. In the case of pure translation, D will be the identity matrix and thus

$$\delta = p + t \tag{6.16}$$

Because of this, the case of pure translation is computationally simpler and thus preferable due the higher frame rates typically found in video data. Since the motion between adjacent frames of standard video is generally quite small, it turns out that setting the deformation matrix to identity is a safe computation [50], leaving us with the translation vector being exactly the displacement vector. A complete description of the tracking equations and feature tracking criteria can be found in Chapter 2, section 11.

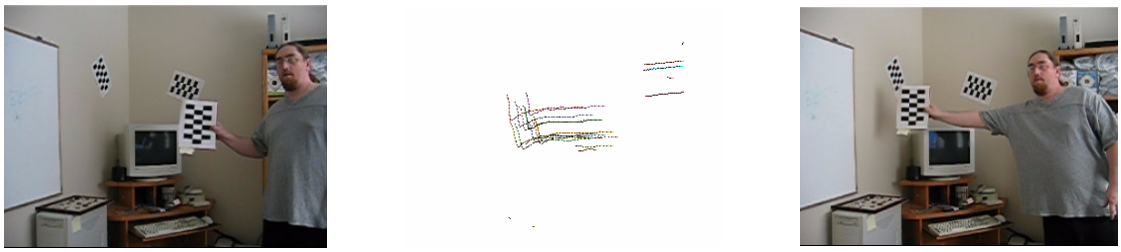


FIGURE 6.7: INITIAL IMAGE, TRAJECTORY IMAGE
(target and head features) tracked over 45 frames, final image

The displacement vector is computed using a pyramid of resolutions because processing a high resolution image is computationally intense. The multi-resolution pyramid within the feature tracker reduces the resolution of the entire image, say by a factor of 2. Tracking occurs by tracking a features general area in the lowest resolution

and upgrading the search for the exact location as it progresses back up the pyramid to the highest resolution.

While tracking features it is possible that an extremely large object motion between frames does occur and features cannot be tracked any further resulting in smaller object trajectories. This is especially true in the case of slower frame rates. So long as the object trajectories overlap in time, the length of the trajectory bears little relevance, although longer sequences help in finding inflection points and help to ensure that trajectory correspondences exist. In Figure 6.7, we have a trajectory image for 3 seconds of a video sequence along with the first and last frames from the from the trajectory sequence.

FIGURE 6.8: TRAJECTORY IMAGE (ENLARGED) FOR A 3 SECOND INTERVAL.

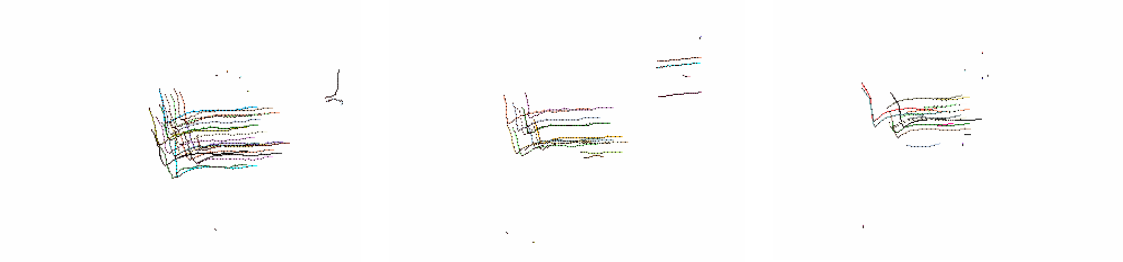
In Figure 6.8 we present an enlarged version of the trajectory image. The image maintains a separate colour for each point feature tracked and specifies the exact point feature position in black for each frame.

6.3.3 Gross approximation of synchronization via trajectory images

We begin by performing a gross approximation of the full frame synchronization. Because we are not assuming trajectory correspondence, we must have enough interest points tracked to ensure correspondence between the 3 views. This will result in cluttered trajectory images, however we can reduce the trajectory images in the presence of inflection points.

An inflection point is found examining the trajectories for similarities in overall shape. In the presence of object motion where direction is changed suddenly; the trajectories show this change at a very obvious point shown in Figure 6.9. In practice this al-

lows us to get within a few frames of correct synchronization, but is never guaranteed to be exact. The reason for this is due to differing frames rates combined with perspective distortions of the fluidly moving objects causing a many-to-one, frame-to-pixel location of inflection points in the trajectory images.



a **b** **c**

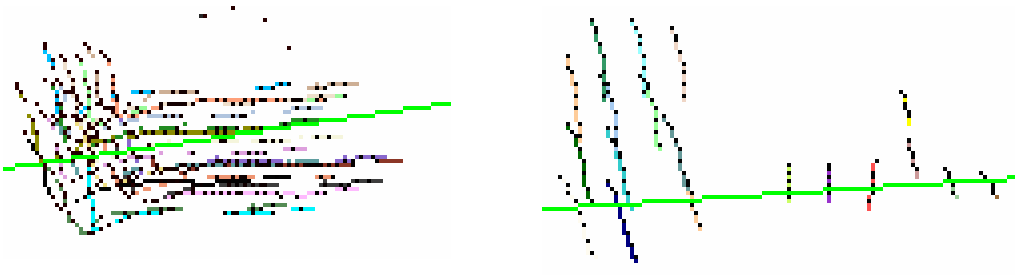
FIGURE 6.9: 3 TRAJECTORY IMAGES WITH OBVIOUS COINCIDENT POINTS OF INFLECTION

In practice, the presence of obvious inflection points may be quite difficult to find, especially when the motions of the dynamic objects are not under control of the application. Furthermore, the trajectories of non rigid objects have different times in which the change of motion represents itself. For example, the loose fitting clothing of a basketball player making a jump shot continues moving upwards after the player has reached the apex of the jump causing the abrupt change in motion of the player's head and clothing to occur in different features at different points in time. In order to resolve these discrepancies, it would be necessary to first solve the trajectory correspondence problem.

A closer examination of motion trajectories and the corresponding frames in the video sequences helps to show how the gross approximation errors occur. When object motion changes, especially if it is in the direction of the optical axis, there are several frames associated to a single pixel location. Furthermore, should the object motion be extremely slow, there are multiple frames associated with the feature location and therefore a higher error in the gross approximation will result. To confirm, we tracked a target over 15 frames as it moved towards a camera along the optical axis. The result was a many to one, frame to pixel location association that made exact localization of the frames impossible.

Once we have identified the inflection points, and implicitly the gross approximation of the synchronization, it is further refined by creating a reduced trajectory image

around the point of inflection and a geometric consensus stage is applied. In cases where inflection points cannot be reliably found, the gross approximation stage can be omitted and we use the larger trajectory images in the consensus stage. This will result in many more consensus trials being performed, as we see in Figure 6.10a, because the epipolar line will potentially intersect with many more trajectories causing the candidate set to be larger. In Figure 6.10b we show a reduced trajectory image (enlarged for viewing) for an 8 frame track after the point of inflection.



a **b**

FIGURE 6.10: EPIPOLAR LINE AND TRAJECTORIES.
The intersection of trajectories and the epipolar line make up the candidate set of points. (a) large trajectory (b) reduced trajectory

6.3.4 Refinement via maximal geometric consensus

During the creation of the trajectory images, we associate a list of frame numbers to each tracked pixel position of the dynamic objects in the trajectory image. We can now effectively compute the synchronization to sub-frame accuracy using the camera geometry and the trajectory images. We do this by selecting a point x in any trajectory in the first image. We then compute the epipolar line that will intersect the corresponding trajectory in the second trajectory image. The epipolar line will also cross other trajectories in the second image, and we use the intersections of the epipolar line and the trajectories to create a candidate set of matching points. As shown in Figure 6.11, these candidate frames can be computed to sub-frame accuracy as the intersection of the trajectory line joining two point positions in adjacent frames. For each point in the candidate set, tensor transfer is applied along with the first point to compute a third point in the third trajectory image. The computed 3rd point (via tensor transfer) is used to find the closest trajectory point. This closest point is also computed to sub frame accuracy as shown in Figure 6.11.

Once the 3 points have been associated to their respective trajectories, the nearest full frames are selected for a consensus trial. We compare a window around the exact tracked point position for each of points using normalized cross correlation to determine whether or not the three points are similar enough to perform the consensus trial. When the points agree, we then verify that a variety of features in the selected frames support the given geometry by generating corner features and performing matching that is guided by the pre-computed geometries. The putative synchronized frame set with the highest consensus overall is selected as the synchronized CCS.

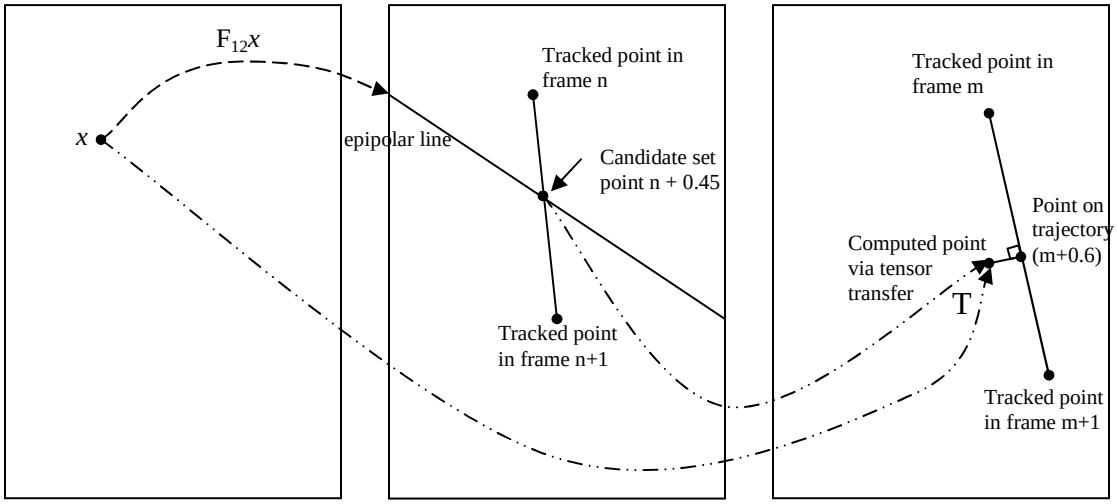


FIGURE 6.11: COMPUTING FRAME VIA EPIPOLAR AND TENSOR TRANSFER.

In practice, one should always start with the slowest camera when selecting the first point *x* because it will help to reduce the number of potential consensus trials as the two slowest cameras define the largest possible error in synchronization time as discussed in section 2. Moreover, attempts should be made to use a reduced trajectory image so that the number of many-to-one frame-to-pixel associations is minimized and therefore the size of the candidate set of matching points will be smaller. If any of the computed points lie on overlapping tracks, we test all the possible combinations of frames. For a point *x* whose epipolar line intersects X trajectories and the subsequent tensor transfer is equidistant to Y trajectories results in a consensus trial for X·Y frame triplets, a drastic reduction in consensus trials compared to (N-2)max_offset^3 which is the number of trials for a smart brute force method as discussed in section 3.

6.3.5 Synchronization in the face of erroneous geometries.

In the presence of error in the computed geometries, an exact answer cannot be trusted. Even a single pixel displacement of the epipolar line will result in an incorrect location of the intersection of trajectories and the epipolar line, which will result in an inexact time localization. In the presence of larger inaccuracies, it is beneficial to examine a broader range of frames when operating our consensus trials. We do this by modifying the generation of the candidate set to include multiple frames from each track that intersect with the epipolar line. We examine a number (ε) of complete frames on either side of the epipolar line. The value for ε is dictated by our confidence in the computed geometry, its error and the distance between tracked points. This will increase the number of consensus trials by a factor of $(2\varepsilon+1)^2$ times, the number of trials where we have absolute confidence in the computed geometries. The optimal epsilon is function of the frame rates and guarantees us to search at least one primary synchronization point. For each sequence, *i*, epsilon is:

$$\varepsilon = \left\lceil \frac{\lambda}{2\rho_i} \right\rceil \tag{6.17}$$

This will result in $(N-2)(2\varepsilon+1)^2$ consensus trials. For our experimental trials, we set ε to be 2 as computed from (6.17).

6.3.6 Algorithm Review: Synchronize

We quickly review the synchronization algorithm

Input: 3 video camera view sequences
Output: frame numbers for 3 synchronized frames

External Functions:
Projective_Transfer(p1,p2) performs tensor transfer between p1,p2 returns point p3
Xcorr(f1,p1,f2,p2,f3,p3) performs pair wise normalized cross correlation for 3 points (*px*) in 3 frames (*fx*)and returns the minimum correlation of all 3 pairings.

1 Select frames that a form static-CCS
2 Compute fundamental matrices(F_{12} and F_{23}) & triliner tensor(T_{123})
3 Generate trajectory images at 3 second intervals
4 Find and match inflection points in trajectory images
5 Generate reduced trajectory images around inflection points
6 Select a point x from $_{f1}$ on a trajectory image 1
7 $l = F_{12}x$ (compute Epipolar line)

```
8  Compute the candidate set of points (xci) i.e. the intersections
   of the trajectories and the epipolar line l
9  For each point pair (x, xci) do
10   xt = Projective_Transfer(x, xci)
11   Compute frame number f2 from xci to sub-frame accuracy (Fig 13)
12   Compute frame number f3 from xt and the nearest trajectory to
          sub-frame accuracy (Fig 13)
13   If (Xcorr( f1,x, ⌊f2+0.5⌋,xci, ⌊f3+0.5⌋, xt) < threshold)
14      Perform guided matching using frames ⌊f1+0.5⌋⌊f2+0.5⌋⌊f3+0.5⌋ and
           the geometry computed in (2)
15       If consensus feature count is maximal then
           Save f1, f2, and f3
16 Endfor(9)
17 Return f1, f2, and f3
```

6.4 Experimental Results

We have applied the algorithm to various sequences, both synthetic and those captured by a variety of different cameras of varying quality and frame rates. We compare to known ground truth where possible, while in the other cases, we compare to our hand selected ground truth.

6.4.1 Synthetic Data

In our synthetic data set, we have a series of static 3D points in a variety of positions. Our dynamic 3D points are the vertices of a cube which we move before constructing each frame in our sequence. The frame rates are the same and constant for each generated sequence, and the sequences are perfectly synchronized. This scenario represents a system of cameras with identical frame rates that are full frame synchronized. The offsets were set to be 0, 5 and 10 frames respectively. In this case, the motion of the cube was arranged so that the vertices of the cube projected to a unique pixel after each motion. This resulted in a trajectory image with a one-to-one pixel/frame number association. The trifocal tensor was derived from the projection matrices and the fundamental matrices were subsequently derived from the tensor. The trajectory images were generated using the projected positions of the 3D vertices of the cube. Due to the simplistic motion, there were no inflection points in the trajectory image, thus application of the consensus algorithm was all that was necessary. Under these ideal conditions, the synchronization was computed exactly to be frame deltas 0, 5 and 10 respectively.

In our next synthetic example, we configured the system to have the same constant frame rates for each generated sequence. In this example, the sequences are not perfectly synchronized and frame deltas of 0, 5.25 and 10.75 were used to represent a system similar to Figure 3. In this case, the motion of the cube was arranged so that the vertices of the cube projected to a unique pixel after each motion and that for each full frame tick, the points moved exactly 4 pixels. This resulted in a trajectory image with a one-to-one pixel/frame number association and allowed us to easily determine the sub-frame offsets. Under these conditions, the synchronization was computed exactly to be frame deltas 0, 5.25 and 10.75 respectively.

6.4.2 Real Data

In the following experiments, we used various digital cameras with video capture capabilities. The cameras had different capture capabilities such as frame rates and resolutions. In our first experiment, we used a system of 3 cameras that grabbed frames on a synchronized basis, we then offset the video sequences by 5 and 10 frames for the second and third cameras respectively to be used as our ground truth. This scenario represents a system of cameras with identical frame rates that are full frame synchronized and the capture process was started simultaneously (as in Figure 1). As we can see in Table 1 the computed full frame synchronization is correct, however due to minor errors in computed geometry, the exact synchronization exhibits the minor errors. The error falls well within the expected maximum error of ½ a frame.

Camera	Gross Approximation	Exact Sync	First Primary Sync Point	Exact Time *Ei* (s)	Universal Time Shift *Si* (s)	Exact First Full Frame	Ground Truth
1	141	141	0	0	1.994	0	0
2	146	146.05	5.05	1.010	0.984	5	5
3	151	150.97	9.97	1.994	0	10	10

Table 1: Synchronization Results

In Figure 12, we show the synchronized frames in rows and sequences in columns for this example. As you can see the events (specifically the hand) are well matched in time.

In our next set of experiments, we utilize 3 off the shelf digital cameras with video capture capabilities. The cameras were of varying quality and frame rates. The camera limitations for this set of experiments are presented in Table 2. This reflects the situation depicted in Figure 3. Given the frame rates, we can expect full frame synchronization to fall within 0.033 seconds of the perfectly accurate synchronization.

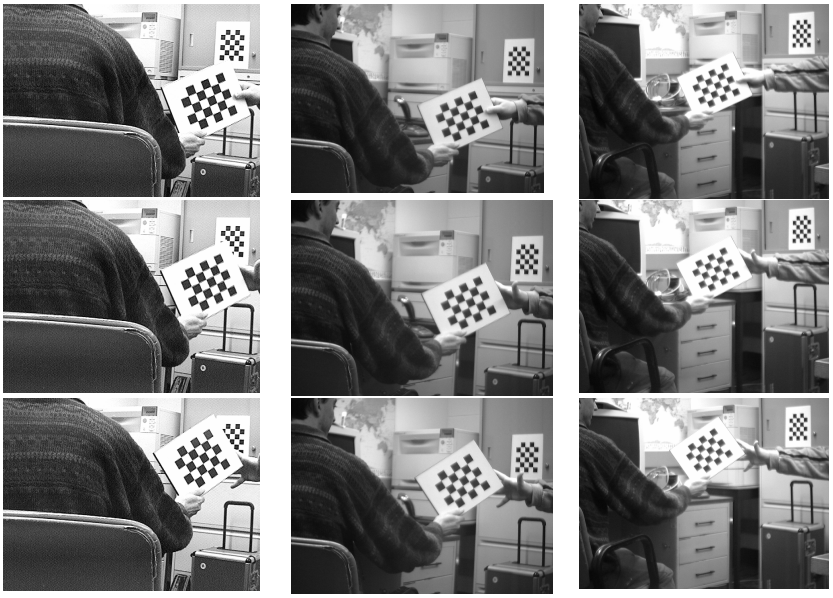


FIGURE 6.12: 3 SYNCHRONIZED (ROWS), CONSECUTIVE FRAMES (COLUMNS)

Camera	Frame Rate	Noise level	Resolution (Sharpness)
1	**15**	**typical**	**low**
2	**15**	**typical**	**standard**
3	**10**	**high**	**standard**

Table 2: Camera Characteristics

Camera	Frame Rate	Gross Approx - imation	Time (s)	Universal Time Shift (s)	Exact Sync	Exact Time (s)	Universal Time Shift (s)	Nearest Full Frame	Time
1	1/15	127	8.467	7.533	126.75	8.45	7.45	127	8.467
2	1/15	228	15.2	0.80	229.33	15.289	0.611	229	15.267
3	1/10	160	16	0	159	15.9	0	159	15.9

Table 3: Synchronization Results

In the first two examples, we used a target as the moving object in order to assure corresponding points would produce corresponding trajectories. In both of these examples, the target was moved such that an obvious change of direction (inflection points)

occurred. These very sharp inflection points allow the gross approximation method to achieve very close results to the synchronized frames with maximal consensus. We can see in Table 3 that the gross approximation in the presence of inflection points are accurate to within a few frames of the exact full frame synchronization. In Table 4, we confirm that the time difference falls within the expected values given the ground truth.

The targets, while helpful in ensuring the accuracy by allowing multiple corresponding trajectories causes the algorithm to force many more consensus trials because of the feature proximity. With fewer corresponding features being tracked, there are fewer points in the candidate sets and thus fewer geometric consensus trials. However, in order to ensure accurate synchronization, we need to ensure that at least one corresponding feature is sufficiently tracked in all 3 cameras sequences.

Ground Truth (user selected)	Time	Difference from computed time
127	8.467	0.017
229	15.267	0.022
159	15.9	0

Table 4: Synchronization Results

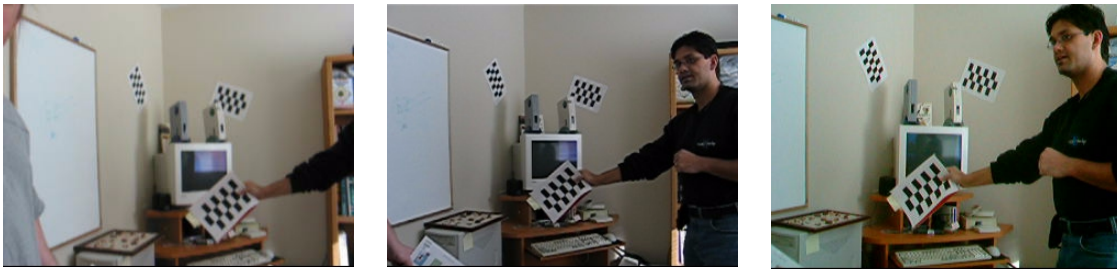


FIGURE 6.13: SELECTED SYNCHRONIZED FRAMES

A second example using the same cameras outlined in Table 2 also bring us to the same conclusion. The results, outlined in Table 5, support the previous experiments findings that the gross approximation, in the presence of inflection points is accurate to within a few frames.

In order for the exact computation of synchronization via geometric consensus to be effective, there is the requirement of corresponding features being successfully tracked. The targets, seen in Figure 13 help to ensure that corresponding features are indeed tracked. As a result, the trajectory images are quite feature-rich. In contrast, without the

use of targets, the trajectory images may be quite sparse due to the static background features being selected automatically over the moving object features. In our final example, we abandon the use of targets and look to automatically track features on dynamic objects.

	Gross Approximation	Exact Synchronization	Full Frame Sync.	Full Frame-Ground Truth
Camera 1 frame	165	167	167	167
Camera 2 frame	212	213.66	214	214
Camera 3 frame	170	170.80	171	170
Total Frame Error	4	1.13	1	N/A

Table 5: Synchronization Results

Unlike our target based approach, the gross approximation became more difficult and required minor manual interventions. Because the features on the moving objects lack the contrast of the target, it was often the case that features on the moving objects would not be automatically selected into the tracked features list. A minor manual step to force feature selection in selected areas helped to generate better trajectory images. Background subtraction techniques would help to remove the need for this manual requirement.

Camera	Frame Rate	Gross Approx - imation	Time (s)	Universal Time Shift (s)	Exact Sync	Exact Time (s)	Universal Time Shift (s)	Nearest Full Frame	Time
1	1/15	244	16.267	11.133	242.80	16.187	11.213	243	16.20
2	1/15	302	20.133	7.267	301.50	20.100	7.300	302	20.13
3	1/10	274	27.400	0	274	27.400	0	274	27.40

Table 6: Synchronization Results

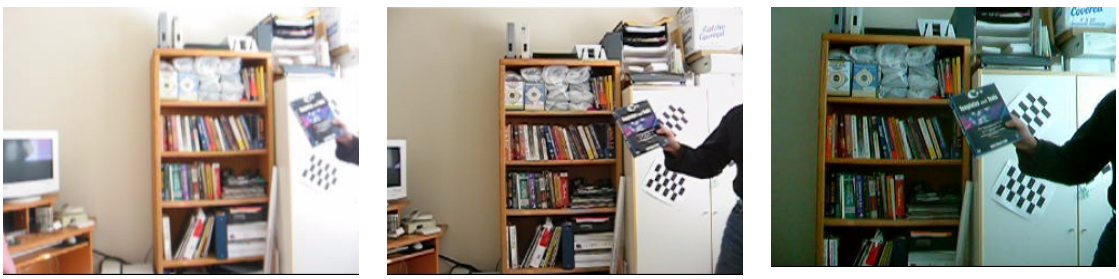


FIGURE 6.14: SELECTED SYNCHRONIZED FRAMES

Again, the results shown in Figure 14, listed in Table 6 and Table 7 fall within the expected maximum error. However, in this specific case, the epipolar line fell exactly halfway between the two points in sequence 2. Our decision to move up, rather than down affected the accuracy and caused a single frame error in the computed full frame synchronization, but does not change the accuracy of the exact computation. A minor anomaly worth noting, is that in this example, the hand selected ground truth value for sequence 2 (301) may be incorrect. All three methods, gross approximation, exact and robust all agree with a value of 302. When selecting the ground truth, we had several people examine the frames and come to a consensus of the frame numbers that they believed were synchronized.

Ground Truth (user selected)	Time	Difference from exact computed time
243	16.200	0
301	20.067	0.033
274	27.400	0

Table 7: Synchronization Results

Example	Camera	Exact with inaccurate geometry	Robust	Ground Truth	Nearest full frame, with accurate geometry
	1	125.33	127	127	127
1	2	227	229	229	229
	3	159	159	159	159
	1	166.25	167	167	167
2	2	211.5	213	214	214
	3	170	170	170	171
	1	240.66	243	243	243
3	2	304.25	302	301	302
	3	274	274	274	274

Table 8: Comparison of methods in the face of geometric inaccuracy

In our final set of experiments, we artificially added error to the computed geometries to simulate degenerate geometries. We then ran the examples again using the robust strategy outlined in section 3.5 for dealing with erroneous geometries. As we can see in

Table 8, the strategy of sliding up and down the trajectories and performing more consensus trails on all the local combinations is helpful in the face of inexact geometries. However, it requires substantially many more consensus trials, and therefore requiring more computing time.

By using the robust method, we are able to achieve results that are very close to ground truth.

6.5 Practical Considerations

During the investigation, several practical issues were raised. Instrumental to the success of the accurate computation of the synchronization is an accurate computation of the camera geometries. The problem of computing an accurate set of camera geometries is considered difficult; and inaccurate geometries, even within a few pixels, can result in an incorrect selection of synchronized frames that are no better than the gross approximation stage. In practice, SIFT features [111] helped to generate accurate geometries in difficult circumstances.

The primary area of concern for the tracking aspect is the avoidance of multiple frame associations to a single feature location and that corresponding features are tracked for some period of time. In order to avoid multiple frame associations, the maximum length of the trajectories should be less than the tracked features intra frame pixel disparity when generating the reduced trajectory images. It is difficult to ensure that corresponding moving object features are tracked in all views, but in practice, targets ensured that many corresponding trajectories existed. However, in situations where targets were not used, simply selecting areas to automatically select features to track helped to improve the situation at the cost of taking away from the hands off approach that the targets allowed. Restricting the area for detection of features to track in an automated method would help to make the algorithm more practical.

Open problems include automatic detection of corresponding inflection points and the automatic detection of corresponding trajectories apriori. While these values are implicitly computed by the method, thus known apostori, knowing them apriori would result

in a reduction of the number of times the consensus step (the largest consumption of time) is required.

6.6 Conclusions and Discussion

In this work we present a novel method for multiple video temporal synchronization using feature tracking and geometric consensus. The proposed method allows for the least constraints being placed on the camera setup and the scene being viewed. The method provides two levels of accuracy by using a two step process of grossly approximating the frame synchronization followed by a refinement step that examines the selected frames for their consensus with the camera geometry. The method has been successfully used on both synthetic data and real data with substantial noise, differing frame rates and varying levels of initial synchronization. Even in the presence of erroneous geometries, it is possible to get very close synchronization results at the cost of performing more consensus trials to account for the geometric inaccuracies.

Chapter 7

Conclusions and Open Problems

7.1 Summary

Video data presents a variety of problems for computing due to introduction of issues not often found in classical computer vision. Primarily, the sheer volume of the data itself is a primary concern that will often prevent the straight forward application of standard computer vision algorithms to each individual frame. As well, video data introduces another dimension into the computation models: time. Temporal constraints also prevent the application of standard algorithms as they are often presented as single image algorithms. For example: content based image retrieval techniques cannot be simply applied to video data. Furthermore, video data is often accompanied by audio data. While not examined directly in this thesis, audio data is yet another dimension for consideration when processing video data. For all of the above reason, the field of computation video has started to emerge as an interesting and necessary field of computer science. As computational video is an emerging research area, this thesis has presented practical algorithms for a variety of problems commonly encountered by applications that utilize video data.

Chapter 3 presented a method to select appropriate frames from a video sequence for subsequent processing using computer vision algorithms. Furthermore, the chapter presents a publicly available platform called the Projective Vision Toolkit that allows future researchers to reduce the learning curve and accelerate research into the field.

Chapter 4 presented a computer vision based approach to the problem of segmenting commercial video. Shots detection forms the cornerstone of many content based video/image retrieval systems. The quest for a perfect segmentation algorithm still remains. However, this chapter presented an improved methodology that significantly outperforms existing techniques.

Chapter 5 presented a Genetic Algorithm based system to perform autocalibration of cameras. Current techniques are not scalable to the volume of data present in video

sequences and thus are not a tractable solution. We presented an extremely fast and accurate method for self calibration of video cameras.

Chapter 6 presented a theoretical look at the mathematical properties of the video synchronization problem and shows the existence of two distinctly different flavors of the synchronization problem. The chapter proceeds to present a solution to the synchronization problem that, unlike existing solutions, considers both flavors of the problem and is not constrained by the presence of large planar surfaces.

The solutions presented in this thesis represent the only the tip of the iceberg when dealing with video data and ring true to the words of Anton Checkov: "*Yet it was clear … that the end was still far, far off, and that the hardest and most complicated part was only just beginning"*. We next examine a list of open problems and future work.

7.2 Future Work

Finally, areas where future work would be beneficial are plenty in emerging fields. We present some areas related to the each of the chapters, but left unexplored by this thesis.

First, corner features used in the Projective Vision Toolkit suffer under certain camera motions when trying to compute corresponding features between views. The same problems that exist for feature matching also exist for feature tracking. An examination into scale invariant features (SIFT) [111] has shown some initial promise within the PVT and in a feature tracking context. More work is necessary to fully develop the idea of Scale Invariant Feature Transform Tracking ($SIFT^2$). This will allow more accurate computation of geometries and more robust feature tracking.

A second area to examine is in video tracking as well. Independently moving objects will temporarily occlude different static areas of the scene as they move causing feature loss in tracking applications. Identifying such occlusions would help to enumerate the independently moving objects and allow re-tracking once the occlusions a gone. Research into detecting and combining occlusions as part of the annotation and tracking of objects in video sequences would prove beneficial because independently moving objects

often possess cohesive features such as color and texture, object segmentation and tracking may prove fruitful.

A third area where further effort is required is the adaptation of the methods in Chapter 5 to allow the autocalibration of video cameras with varying intrinsic parameters. Auto-focus features are becoming standard features on commercial video cameras and in order to provide practical application to the more modern version of video cameras, it will be necessary to adapt the cost functions presented in Chapter 5 to minimally allow for differing focal lengths between views.

Finally, there are a variety of future problems that arise due simply to the solutions of the problems presented in this thesis. Real-time constraints are always a consideration for applications such as real-time television viewing by computer programs, autonomous vehicle projects where real-time reaction is necessary and real-time environment recreation for virtual reality applications. Adaptations to color information also present an interesting set of problems. As this thesis performed the majority of its work in the luminance domain, a detailed investigation into the use of color to perform tasks may also prove fruitful.

Appendix A

Exact Match counts for PVT Example Sets

Image pair and Location	Correlation Matches	Filtered Matches	% difference	Fundamental Support Matches	% difference from filtered	% difference from Correlation
ex1/bighouse/c-000101.pgm-c-000102.pgm.matches	417	389	6.71	363	6.68	12.95
ex1/bighouse/c-000102.pgm-c-000103.pgm.matches	349	321	8.02	313	2.49	10.32
ex1/bighouse/c-000103.pgm-c-000104.pgm.matches	470	447	4.89	420	6.04	10.64
ex1/bighouse/c-000104.pgm-c-000105.pgm.matches	487	479	1.64	455	5.01	6.57
ex1/bighouse/c-000105.pgm-c-000106.pgm.matches	393	373	5.09	366	1.88	6.87
ex1/bighouse/c-000106.pgm-c-000107.pgm.matches	531	530	0.19	518	2.26	2.45
ex1/bighouse/c-000107.pgm-c-000108.pgm.matches	422	405	4.03	393	2.96	6.87
ex1/bighouse/c-000108.pgm-c-000109.pgm.matches	509	506	0.59	494	2.37	2.95
ex1/bighouse/c-000109.pgm-c-000110.pgm.matches	505	504	0.20	487	3.37	3.56
ex1/bighouse/c-000110.pgm-c-000111.pgm.matches	433	411	5.08	398	3.16	8.08
ex1/chapel/p0000888.jpg-p0000889.jpg.matches	299	210	29.77	199	5.24	33.44
ex1/chapel/p0000889.jpg-p0000890.jpg.matches	299	196	34.45	165	15.82	44.82
ex1/chapel/p0000890.jpg-p0000891.jpg.matches	289	255	11.76	234	8.24	19.03
ex1/chapel/p0000891.jpg-p0000892.jpg.matches	299	219	26.76	200	8.68	33.11
ex1/chapel/p0000892.jpg-p0000893.jpg.matches	303	254	16.17	238	6.30	21.45
ex1/chapel/p0000893.jpg-p0000894.jpg.matches	265	179	32.45	164	8.38	38.11
ex1/chapel/p0000894.jpg-p0000895.jpg.matches	291	174	40.21	168	3.45	42.27
ex1/chapel/p0000895.jpg-p0000896.jpg.matches	321	251	21.81	228	9.16	28.97
ex1/chapel/p0000896.jpg-p0000897.jpg.matches	283	176	37.81	159	9.66	43.82
ex1/chapel/p0000897.jpg-p0000898.jpg.matches	296	254	14.19	189	25.59	36.15
ex1/chapel/p0000898.jpg-p0000899.jpg.matches	282	236	16.31	208	11.86	26.24
ex1/chapel/p0000899.jpg-p0000900.jpg.matches	293	162	44.71	149	8.02	49.15
ex1/climber/p0000361.jpg-p0000362.jpg.matches	243	132	45.68	73	4.70	9.96
ex1/climber/p0000362.jpg-p0000363.jpg.matches	229	131	42.79	74	4.51	6.69
ex1/climber/p0000363.jpg-p0000364.jpg.matches	242	139	42.56	94	3237	6116
ex1/climber/p0000364.jpg-p0000365.jpg.matches	247	128	48.18	88	31.5	64.7
ex1/climber/p0000365.jpg-p0000366.jpg.matches	225	113	49.78	79	30.0	64.8
ex1/climber/p0000366.jpg-p0000367.jpg.matches	239	101	57.74	63	37.62	73.64
ex1/climber/p0000367.jpg-p0000368.jpg.matches	230	115	50.00	79	31.30	65.65
ex1/climber/p0000368.jpg-p0000369.jpg.matches	245	119	51.43	89	25.21	63.67
ex1/climber/p0000369.jpg-p0000370.jpg.matches	238	143	39.92	85	40.56	64.29
ex1/climber/p0000370.jpg-p0000371.jpg.matches	251	107	57.37	76	28.97	69.72
ex1/climber/p0000371.jpg-p0000372.jpg.matches	240	126	47.50	87	30.95	63.75
ex1/climber/p0000372.jpg-p0000373.jpg.matches	243	126	48.15	88	30.16	63.79
ex1/equiproom/p0001989.jpg-p0001990.jpg.matches	106	103	2.83	20	80.58	81.13
ex1/equiproom/p0001990.jpg-p0001991.jpg.matches	209	182	12.92	107	41.21	48.80
ex1/equiproom/p0001991.jpg-p0001992.jpg.matches	207	168	18.84	119	29.17	42.51
ex1/equiproom/p0001992.jpg-p0001993.jpg.matches	284	254	10.56	205	19.29	27.82
ex1/equiproom/p0001993.jpg-p0001994.jpg.matches	259	231	10.81	157	32.03	39.38
ex1/equiproom/p0001994.jpg-p0001995.jpg.matches	279	207	25.81	183	11.59	34.41

ex1/equiproom/p0001995.jpg-p0001996.jpg.matches	259	219	15.44	168	23.29	35.14
ex1/equiproom/p0001996.jpg-p0001997.jpg.matches	286	266	6.99	221	16.92	22.73
ex2/castle/kasteel101.ppm-kasteel102.ppm.matches	122	121	0.82	42	65.29	65.57
ex2/castle/kasteel102.ppm-kasteel103.ppm.matches	433	425	1.85	400	5.88	7.62
ex2/castle/kasteel103.ppm-kasteel104.ppm.matches	128	91	28.91	91	0.00	28.91
ex2/castle/kasteel104.ppm-kasteel105.ppm.matches	74	52	29.73	40	23.08	45.95
ex2/castle/kasteel105.ppm-kasteel106.ppm.matches	166	139	16.27	132	5.04	20.48
ex2/etlueshiba/etl101.pgm-etl102.pgm.matches	184	183	0.54	115	37.16	37.50
ex2/etlueshiba/etl102.pgm-etl103.pgm.matches	151	149	1.32	95	36.24	37.09
ex2/etlueshiba/etl103.pgm-etl104.pgm.matches	148	147	0.68	87	40.82	41.22
ex2/etlueshiba/etl104.pgm-etl105.pgm.matches	75	74	1.33	47	36.49	37.33
ex2/lab4thfloor/p0000748.jpg-p0000749.jpg.matches	302	241	20.20	143	40.66	52.65
ex2/lab4thfloor/p0000749.jpg-p0000750.jpg.matches	260	201	22.69	106	47.26	59.23
ex2/lab4thfloor/p0000750.jpg-p0000751.jpg.matches	305	269	11.80	171	36.43	43.93
ex2/lab4thfloor/p0000751.jpg-p0000752.jpg.matches	320	292	8.75	182	37.67	43.13
ex2/lab4thfloor/p0000752.jpg-p0000753.jpg.matches	262	184	29.77	101	45.11	61.45
ex2/lab4thfloor/p0000753.jpg-p0000754.jpg.matches	250	187	25.20	79	57.75	68.40
ex2/lab4thfloor/p0000754.jpg-p0000755.jpg.matches	314	282	10.19	139	50.71	55.73
ex2/lab4thfloor/p0000755.jpg-p0000756.jpg.matches	259	197	23.94	71	63.96	72.59
ex2/lab4thfloor/p0000756.jpg-p0000757.jpg.matches	254	223	12.20	94	57.85	62.99
ex2/lab4thfloor/p0000757.jpg-p0000758.jpg.matches	238	223	6.30	65	70.85	72.69
ex3/csroom/p0000827.jpg-p0000828.jpg.matches	210	172	18.10	85	50.58	59.52
ex3/csroom/p0000828.jpg-p0000829.jpg.matches	217	166	23.50	84	49.40	61.29
ex3/csroom/p0000829.jpg-p0000830.jpg.matches	145	131	9.66	34	74.05	76.55
ex3/csroom/p0000830.jpg-p0000831.jpg.matches	234	142	39.32	96	32.39	58.97
ex3/csroom/p0000831.jpg-p0000832.jpg.matches	264	225	14.77	120	46.67	54.55
ex3/csroom/p0000832.jpg-p0000833.jpg.matches	241	149	38.17	97	34.90	59.75
ex3/csroom/p0000833.jpg-p0000834.jpg.matches	286	201	29.72	145	27.86	49.30
ex3/csroom/p0000834.jpg-p0000835.jpg.matches	267	182	31.84	90	50.55	66.29
ex3/readingroom/p0000842.jpg-p0000843.jpg.matches	176	137	22.16	81	40.88	53.98
ex3/readingroom/p0000843.jpg-p0000844.jpg.matches	219	171	21.92	129	24.56	41.10
ex3/readingroom/p0000844.jpg-p0000845.jpg.matches	249	211	15.26	163	22.75	34.54
ex3/reidsculpt/p0001070.jpg-p0001071.jpg.matches	345	287	16.81	157	45.30	54.49
ex3/reidsculpt/p0001071.jpg-p0001072.jpg.matches	465	423	9.03	216	48.94	53.55
ex3/reidsculpt/p0001072.jpg-p0001073.jpg.matches	329	259	21.28	121	53.28	63.22
ex3/reidsculpt/p0001073.jpg-p0001074.jpg.matches	378	331	12.43	145	56.19	61.64
ex3/reidsculpt/p0001074.jpg-p0001075.jpg.matches	320	251	21.56	119	52.59	62.81
ex3/reidsculpt/p0001075.jpg-p0001076.jpg.matches	405	375	7.41	184	50.93	54.57
ex3/reidsculpt/p0001076.jpg-p0001077.jpg.matches	384	346	9.90	156	54.91	59.38
ex3/totem1/p0001062.jpg-p0001063.jpg.matches	358	269	24.86	233	13.38	34.92
ex3/totem1/p0001063.jpg-p0001064.jpg.matches	327	241	26.30	186	22.82	43.12
ex3/totem1/p0001064.jpg-p0001065.jpg.matches	334	256	23.35	218	14.84	34.73
ex3/totem1/p0001065.jpg-p0001066.jpg.matches	363	281	22.59	246	12.46	32.23
ex3/totem1/p0001066.jpg-p0001067.jpg.matches	372	297	20.16	252	15.15	32.26
ex3/totem1/p0001067.jpg-p0001068.jpg.matches	307	235	23.45	202	14.04	34.20
ex3/totem1/p0001068.jpg-p0001069.jpg.matches	341	266	21.99	215	19.17	36.95
ex4/workshop1/p0001669.jpg-p0001670.jpg.matches	202	122	39.60	87	28.69	56.93
ex4/workshop1/p0001670.jpg-p0001671.jpg.matches	217	140	35.48	97	30.71	55.30

ex4/workshop1/p0001671.jpg-p0001672.jpg.matches	232	179	22.84	97	45.81	58.19
ex4/workshop1/p0001672.jpg-p0001673.jpg.matches	282	242	14.18	157	35.12	44.33
ex4/workshop1/p0001673.jpg-p0001674.jpg.matches	353	336	4.82	239	28.87	32.29
ex4/workshop1/p0001674.jpg-p0001675.jpg.matches	312	272	12.82	190	30.15	39.10
ex4/workshop1/p0001675.jpg-p0001676.jpg.matches	290	237	18.28	166	29.96	42.76
ex4/workshop2/p0001677.jpg-p0001678.jpg.matches	175	169	3.43	42	75.15	76.00
ex4/workshop2/p0001678.jpg-p0001679.jpg.matches	229	182	20.52	108	40.66	52.84
ex4/workshop2/p0001679.jpg-p0001680.jpg.matches	201	158	21.39	63	60.13	68.66
ex4/workshop2/p0001680.jpg-p0001681.jpg.matches	210	163	22.38	95	41.72	54.76
ex4/workshop2/p0001681.jpg-p0001682.jpg.matches	245	216	11.84	102	52.78	58.37
ex4/workshop2/p0001682.jpg-p0001683.jpg.matches	214	135	36.92	70	48.15	67.29
ex4/workshop2/p0001683.jpg-p0001684.jpg.matches	211	156	26.07	81	48.08	61.61
ex4/workshop2/p0001684.jpg-p0001685.jpg.matches	292	271	7.19	157	42.07	46.23
ex4/workshop3/p0001686.jpg-p0001687.jpg.matches	198	192	3.03	66	65.63	66.67
ex4/workshop3/p0001687.jpg-p0001688.jpg.matches	179	167	6.70	43	74.25	75.98
ex4/workshop3/p0001688.jpg-p0001689.jpg.matches	157	148	5.73	39	73.65	75.16
ex4/workshop3/p0001689.jpg-p0001690.jpg.matches	154	152	1.30	39	74.34	74.68
ex4/workshop3/p0001690.jpg-p0001691.jpg.matches	159	158	0.63	28	82.28	82.39
ex4/workshop3/p0001691.jpg-p0001692.jpg.matches	173	166	4.05	37	77.7	78.61
ex4/workshop3/p0001692.jpg-p0001693.jpg.matches	202	182	9.90	61	66.48	69.80
ex5/bsmnt/bsmnt100.pgm-bsmnt101.pgm.matches	281	280	0.36	241	13.93	14.23
ex5/bsmnt/bsmnt101.pgm-bsmnt102.pgm.matches	288	286	0.69	263	8.04	8.68
ex5/bsmnt/bsmnt102.pgm-bsmnt103.pgm.matches	291	289	0.69	236	18.34	18.90
ex5/bsmnt/bsmnt103.pgm-bsmnt104.pgm.matches	273	272	0.37	233	14.34	14.65
ex5/bsmnt/bsmnt104.pgm-bsmnt105.pgm.matches	277	276	0.36	240	13.04	13.36
ex5/bsmnt/bsmnt105.pgm-bsmnt106.pgm.matches	254	252	0.79	219	13.10	13.78
ex5/bsmnt/bsmnt106.pgm-bsmnt107.pgm.matches	279	275	1.43	242	12.00	13.26
ex5/bsmnt/bsmnt107.pgm-bsmnt108.pgm.matches	238	232	2.52	145	37.50	39.08
ex5/bsmnt/bsmnt108.pgm-bsmnt109.pgm.matches	229	226	1.31	148	34.51	35.37
ex5/bsmnt/bsmnt109.pgm-bsmnt110.pgm.matches	256	252	1.56	212	15.87	17.19
ex5/montrealstatue/p001756.jpg-p001757.jpg.matches	299	182	39.13	163	10.44	45.48
ex5/montrealstatue/p001757.jpg-p001758.jpg.matches	323	211	34.67	184	12.80	43.03
ex5/montrealstatue/p001758.jpg-p001759.jpg.matches	337	293	13.06	242	17.41	28.19
ex5/montrealstatue/p001759.jpg-p001760.jpg.matches	330	267	19.09	207	22.47	37.27
ex5/vehicle/image109.jpg-image110.jpg.matches	257	141	45.14	136	3.55	47.08
ex5/vehicle/image110.jpg-image111.jpg.matches	209	184	11.96	177	3.80	15.31
ex5/vehicle/image111.jpg-image112.jpg.matches	232	193	16.81	181	6.22	21.98
ex5/vehicle/image112.jpg-image113.jpg.matches	219	189	13.70	173	8.47	21.00
ex5/vehicle/image113.jpg-image114.jpg.matches	143	110	23.08	95	13.64	33.57
ex5/vehicle/image114.jpg-image115.jpg.matches	163	133	18.40	103	22.56	36.81
ex5/vehicle/image115.jpg-image116.jpg.matches	186	162	12.90	127	21.60	31.72
ex5/vehicle/image116.jpg-image117.jpg.matches	136	108	20.59	88	18.52	35.29

Example and location	Putative Triplets	Tensor Support Matches	% difference
ex1/bighouse/c-000101.pgm-c-000102.pgm-c-000103.pgm.matches	185	156	15.68
ex1/bighouse/c-000102.pgm-c-000103.pgm-c-000104.pgm.matches	209	196	6.22
ex1/bighouse/c-000103.pgm-c-000104.pgm-c-000105.pgm.matches	279	236	15.41
ex1/bighouse/c-000104.pgm-c-000105.pgm-c-000106.pgm.matches	269	236	12.27
ex1/bighouse/c-000105.pgm-c-000106.pgm-c-000107.pgm.matches	278	168	39.57
ex1/bighouse/c-000106.pgm-c-000107.pgm-c-000108.pgm.matches	311	243	21.86
ex1/bighouse/c-000107.pgm-c-000108.pgm-c-000109.pgm.matches	310	238	23.23
ex1/bighouse/c-000108.pgm-c-000109.pgm-c-000110.pgm.matches	378	353	6.61
ex1/bighouse/c-000109.pgm-c-000110.pgm-c-000111.pgm.matches	287	225	21.60
ex1/chapel/p0000888.jpg-p0000889.jpg-p0000890.jpg.matches	52	43	17.31
ex1/chapel/p0000889.jpg-p0000890.jpg-p0000891.jpg.matches	61	61	0.00
ex1/chapel/p0000890.jpg-p0000891.jpg-p0000892.jpg.matches	79	52	34.18
ex1/chapel/p0000891.jpg-p0000892.jpg-p0000893.jpg.matches	83	60	27.71
ex1/chapel/p0000892.jpg-p0000893.jpg-p0000894.jpg.matches	61	47	22.95
ex1/chapel/p0000893.jpg-p0000894.jpg-p0000895.jpg.matches	47	33	29.79
ex1/chapel/p0000894.jpg-p0000895.jpg-p0000896.jpg.matches	66	52	21.21
ex1/chapel/p0000895.jpg-p0000896.jpg-p0000897.jpg.matches	58	46	20.69
ex1/chapel/p0000896.jpg-p0000897.jpg-p0000898.jpg.matches	49	34	30.61
ex1/chapel/p0000897.jpg-p0000898.jpg-p0000899.jpg.matches	67	53	20.90
ex1/chapel/p0000898.jpg-p0000899.jpg-p0000900.jpg.matches	44	35	20.45
ex1/climber/p0000361.jpg-p0000362.jpg-p0000363.jpg.matches	21	15	28.57
ex1/climber/p0000362.jpg-p0000363.jpg-p0000364.jpg.matches	26	25	3.85
ex1/climber/p0000363.jpg-p0000364.jpg-p0000365.jpg.matches	25	20	20.00
ex1/climber/p0000364.jpg-p0000365.jpg-p0000366.jpg.matches	20	20	0.00
ex1/climber/p0000365.jpg-p0000366.jpg-p0000367.jpg.matches	18	12	33.33
ex1/climber/p0000366.jpg-p0000367.jpg-p0000368.jpg.matches	20	15	25.00
ex1/climber/p0000367.jpg-p0000368.jpg-p0000369.jpg.matches	23	15	34.78
ex1/climber/p0000368.jpg-p0000369.jpg-p0000370.jpg.matches	20	17	15.00
ex1/climber/p0000369.jpg-p0000370.jpg-p0000371.jpg.matches	21	22	-4.7%
ex1/climber/p0000370.jpg-p0000371.jpg-p0000372.jpg.matches	22	21	4.55
ex1/climber/p0000371.jpg-p0000372.jpg-p0000373.jpg.matches	31	19	38.71
ex1/equiproom/p0001989.jpg-p0001990.jpg-p0001991.jpg.matches	3	2	33.33
ex1/equiproom/p0001990.jpg-p0001991.jpg-p0001992.jpg.matches	18	12	33.33
ex1/equiproom/p0001991.jpg-p0001992.jpg-p0001993.jpg.matches	48	24	50.00
ex1/equiproom/p0001992.jpg-p0001993.jpg-p0001994.jpg.matches	55	33	40.00
ex1/equiproom/p0001993.jpg-p0001994.jpg-p0001995.jpg.matches	58	34	41.38
ex1/equiproom/p0001994.jpg-p0001995.jpg-p0001996.jpg.matches	57	36	36.84
ex1/equiproom/p0001995.jpg-p0001996.jpg-p0001997.jpg.matches	68	38	44.12
ex2/castle/kasteel101.ppm-kasteel102.ppm-kasteel103.ppm.matches	27	19	29.63
ex2/castle/kasteel102.ppm-kasteel103.ppm-kasteel104.ppm.matches	59	47	20.34
ex2/castle/kasteel103.ppm-kasteel104.ppm-kasteel105.ppm.matches	13	12	7.69
ex2/castle/kasteel104.ppm-kasteel105.ppm-kasteel106.ppm.matches	23	22	4.35
ex2/etlueshiba/etl101.pgm-etl102.pgm-etl103.pgm.matches	10	10	0.00
ex2/etlueshiba/etl102.pgm-etl103.pgm-etl104.pgm.matches	17	17	0.00
ex2/etlueshiba/etl103.pgm-etl104.pgm-etl105.pgm.matches	9	9	0.00
ex2/lab4thfloor/p0000748.jpg-p0000749.jpg-p0000750.jpg.matches	38	24	36.84

ex2/lab4thfloor/p0000749.jpg-p0000750.jpg-p0000751.jpg.matches	36	36	0.00
ex2/lab4thfloor/p0000750.jpg-p0000751.jpg-p0000752.jpg.matches	72	35	51.39
ex2/lab4thfloor/p0000751.jpg-p0000752.jpg-p0000753.jpg.matches	47	25	46.81
ex2/lab4thfloor/p0000752.jpg-p0000753.jpg-p0000754.jpg.matches	30	25	16.67
ex2/lab4thfloor/p0000753.jpg-p0000754.jpg-p0000755.jpg.matches	23	23	0.00
ex2/lab4thfloor/p0000754.jpg-p0000755.jpg-p0000756.jpg.matches	20	10	50.00
ex2/lab4thfloor/p0000755.jpg-p0000756.jpg-p0000757.jpg.matches	12	12	0.00
ex2/lab4thfloor/p0000756.jpg-p0000757.jpg-p0000758.jpg.matches	13	13	0.00
ex3/csroom/p0000827.jpg-p0000828.jpg-p0000829.jpg.matches	17	17	0.00
ex3/csroom/p0000828.jpg-p0000829.jpg-p0000830.jpg.matches	7	7	0.00
ex3/csroom/p0000829.jpg-p0000830.jpg-p0000831.jpg.matches	10	10	0.00
ex3/csroom/p0000830.jpg-p0000831.jpg-p0000832.jpg.matches	32	32	0.00
ex3/csroom/p0000831.jpg-p0000832.jpg-p0000833.jpg.matches	27	15	44.44
ex3/csroom/p0000832.jpg-p0000833.jpg-p0000834.jpg.matches	40	24	40.00
ex3/csroom/p0000833.jpg-p0000834.jpg-p0000835.jpg.matches	34	34	0.00
ex3/readingroom/p0000842.jpg-p0000843.jpg-p0000844.jpg.matches	23	22	4.35
ex3/readingroom/p0000843.jpg-p0000844.jpg-p0000845.jpg.matches	65	48	26.15
ex3/reidsculpt/p0001070.jpg-p0001071.jpg-p0001072.jpg.matches	79	38	51.90
ex3/reidsculpt/p0001071.jpg-p0001072.jpg-p0001073.jpg.matches	48	28	41.67
ex3/reidsculpt/p0001072.jpg-p0001073.jpg-p0001074.jpg.matches	47	27	42.55
ex3/reidsculpt/p0001073.jpg-p0001074.jpg-p0001075.jpg.matches	50	31	38.00
ex3/reidsculpt/p0001074.jpg-p0001075.jpg-p0001076.jpg.matches	49	24	51.02
ex3/reidsculpt/p0001075.jpg-p0001076.jpg-p0001077.jpg.matches	52	24	53.85
ex3/totem1/p0001062.jpg-p0001063.jpg-p0001064.jpg.matches	96	64	33.33
ex3/totem1/p0001063.jpg-p0001064.jpg-p0001065.jpg.matches	89	57	35.96
ex3/totem1/p0001064.jpg-p0001065.jpg-p0001066.jpg.matches	120	79	34.17
ex3/totem1/p0001065.jpg-p0001066.jpg-p0001067.jpg.matches	132	78	40.91
ex3/totem1/p0001066.jpg-p0001067.jpg-p0001068.jpg.matches	98	59	39.80
ex3/totem1/p0001067.jpg-p0001068.jpg-p0001069.jpg.matches	86	86	0.00
ex4/workshop1/p0001669.jpg-p0001670.jpg-p0001671.jpg.matches	21	16	23.81
ex4/workshop1/p0001670.jpg-p0001671.jpg-p0001672.jpg.matches	21	14	33.33
ex4/workshop1/p0001671.jpg-p0001672.jpg-p0001673.jpg.matches	29	29	0.00
ex4/workshop1/p0001672.jpg-p0001673.jpg-p0001674.jpg.matches	65	30	53.85
ex4/workshop1/p0001673.jpg-p0001674.jpg-p0001675.jpg.matches	86	86	0.00
ex4/workshop1/p0001674.jpg-p0001675.jpg-p0001676.jpg.matches	67	38	43.28
ex4/workshop2/p0001677.jpg-p0001678.jpg-p0001679.jpg.matches	18	12	33.33
ex4/workshop2/p0001678.jpg-p0001679.jpg-p0001680.jpg.matches	20	15	25.00
ex4/workshop2/p0001679.jpg-p0001680.jpg-p0001681.jpg.matches	14	11	21.43
ex4/workshop2/p0001680.jpg-p0001681.jpg-p0001682.jpg.matches	11	8	27.27
ex4/workshop2/p0001681.jpg-p0001682.jpg-p0001683.jpg.matches	20	16	20.00
ex4/workshop2/p0001682.jpg-p0001683.jpg-p0001684.jpg.matches	11	9	18.18
ex4/workshop2/p0001683.jpg-p0001684.jpg-p0001685.jpg.matches	34	25	26.47
ex4/workshop3/p0001686.jpg-p0001687.jpg-p0001688.jpg.matches	14	14	0.00
ex4/workshop3/p0001687.jpg-p0001688.jpg-p0001689.jpg.matches	5	5	0.00
ex4/workshop3/p0001688.jpg-p0001689.jpg-p0001690.jpg.matches	6	6	0.00
ex4/workshop3/p0001689.jpg-p0001690.jpg-p0001691.jpg.matches	4	2	50.00
ex4/workshop3/p0001690.jpg-p0001691.jpg-p0001692.jpg.matches	1	1	0.00
ex4/workshop3/p0001691.jpg-p0001692.jpg-p0001693.jpg.matches	11	8	27.27
ex5/bsmnt/bsmnt100.pgm-bsmnt101.pgm-bsmnt102.pgm.matches	198	167	15.66
ex5/bsmnt/bsmnt101.pgm-bsmnt102.pgm-bsmnt103.pgm.matches	184	134	27.17

ex5/bsmnt/bsmnt102.pgm-bsmnt103.pgm-bsmnt104.pgm.matches	171	140	18.13
ex5/bsmnt/bsmnt103.pgm-bsmnt104.pgm-bsmnt105.pgm.matches	180	135	25.00
ex5/bsmnt/bsmnt104.pgm-bsmnt105.pgm-bsmnt106.pgm.matches	170	144	15.29
ex5/bsmnt/bsmnt105.pgm-bsmnt106.pgm-bsmnt107.pgm.matches	168	132	21.43
ex5/bsmnt/bsmnt106.pgm-bsmnt107.pgm-bsmnt108.pgm.matches	114	66	42.11
ex5/bsmnt/bsmnt107.pgm-bsmnt108.pgm-bsmnt109.pgm.matches	79	46	41.77
ex5/bsmnt/bsmnt108.pgm-bsmnt109.pgm-bsmnt110.pgm.matches	103	54	47.57
ex5/montrealstatue/p001756.jpg-p001757.jpg-p001758.jpg.matches	90	63	30.00
ex5/montrealstatue/p001757.jpg-p001758.jpg-p001759.jpg.matches	116	84	27.59
ex5/montrealstatue/p001758.jpg-p001759.jpg-p001760.jpg.matches	117	72	38.46
ex5/vehicle/image109.jpg-image110.jpg-image111.jpg.matches	63	45	28.57
ex5/vehicle/image110.jpg-image111.jpg-image112.jpg.matches	96	72	25.00
ex5/vehicle/image111.jpg-image112.jpg-image113.jpg.matches	82	63	23.17
ex5/vehicle/image112.jpg-image113.jpg-image114.jpg.matches	51	41	19.61
ex5/vehicle/image113.jpg-image114.jpg-image115.jpg.matches	37	26	29.73
ex5/vehicle/image114.jpg-image115.jpg-image116.jpg.matches	57	39	31.58
ex5/vehicle/image115.jpg-image116.jpg-image117.jpg.matches	41	27	34.15

Appendix B

The Portable Image Library (PIL)

One of the fallouts from this thesis was the creation of a library to handle a variety of different image and video formats. The Portable Image Library (PIL) provides a consistent interface and allows the accessing and manipulation of still image and video formats for the three most popular computing platforms today: SUN-Solaris, Linux and Microsoft Windows. PIL is a C API (Application Programming Interface) that makes use of its own canonical image format, and allows a programmer to load various image formats such as GIF, JPG, PNG, TIFF, BMP, PGM, PPM, DICOM and PCT. Furthermore, the library supports the frame-by-frame loading of video formats such as MPEG across all platforms and AVI, WMV formats on Windows platforms.

Because PIL allows students to be able to open image and video data of a variety of different formats with only a few lines of C code, students can immediately begin implementing image and video processing algorithms without consideration of the complexities that surround image file formats and byte ordering on different computer processors. As such PIL is an ideal platform to base computer vision and image processing courses on. PIL is freely available to all.

Appendix C

The Projective Vision Toolkit (PVT)

Based on the PIL Libraries, the Projective Vision Toolkit (PVT) is a series of utilities available in binary form that allow you to take an image sequence and compute the fundamental matrix and trifocal tensor. The current version only goes as far as computing these two quantities, along with the correspondences that support them. It does so completely automatically, using only natural features. The most important assumption is that the maximum motion of a single feature is limited (usually to 1/3 of the image size). We are able to process images that are more widely separated than those from a video camera, but can not handle ultra wide separations. If one wishes to perform a reconstruction of the camera positions it is necessary to autocalibrate (or to a-priori have the calibration) which is also provided as part of the PVT. In each of our current examples (outlined in Appendix A) we have a VRML file (.wrl extension) which shows the reconstruction of the camera positions along with the features that were detected. In this case the reconstruction was obtained by sending the correspondences and calibration information to the Photomodeler package.

The PVT comes with a number of utilities that are also useful in the context of research and/or teaching. Geometry and sequence viewers help to solidify the ideas behind epipolar geometry by allowing visual examination of tensor and epipolar transfer.

Bibliography

"If I have seen farther than others, it is because I was standing on the shoulders of giants." - Isaac Newton

[1] Anthony Whitehead. Biometrics and Automated Authentication: A presentation of a Minimal Space Template Authentication Algorithm and in Improved Classification Methodology. Honours Project, 1995

[2] Kuhl, F., "Classification and Recognition of Hand-Printed Characters", *IEE National Convention*, pp. 75-93, March 1963.

[3] R. Bajcsy, "Computer recognition of roads from satellite pictures".*IEEE Transactions on System, Man and Cybernetics*, 6(9), 1976

[4] S. Peleg, B. Rousso, A. Rav-Acha, and A. Zomet, "Mosaicing on Adaptive Manifolds", *IEEE Transactions. On PAMI*, pp. 1144-1154, October 2000.

[5] C. Thorpe and T. Kanade, "Vision and Navigation for the Carnegie Mellon Navlab", *Proceedings of the 1985 DARPA Image Understanding Workshop*, pp. 143-52, 1985.

[6] R. Sukthankar, "RACCOON: A Real-time Autonomous Car Chaser Operating Optimally at Night," *Proceedings of the IEEE IV'93*, 1993.

[7] H.C. Longuet-Higgens. "A computer algorithm for reconstructing a scene from two projections". *Nature*, vol.293:133-135, 1981.

[8] O.D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig?. *Proc. 2nd European Conference on Computer Vision,1992*, pp. 563-578

[9] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. *Proc. 2nd European Conference on Computer Vision,1992*, pp. 579-587

[10] Z. Zhang. "Determining the Epipolar Geometry and Its Uncertainty: A Review" *Technical Report RR-2927*, INRIA, 1996.

[11] P.H.S. Torr and D.W. Murray, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix", *International Journal of Computer Vision*, vol 24 pp 271-300, 1997

[12] A. Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. *International Conference on Computer Vision*, 1995

[13] R. Jain, R. Kasturi, and B. Schunck. Machine Vision. McGraw-Hill and MIT Press, second edition. 1995.

[14] G. Xu, and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition.* Kluwer Academic Publishers. 1996.

[15] O.D. Faugeras. Three Dimensional Computer Vision – A Geometric Viewpoint. MIT Press. 1993

[16] J. Stolfi. *Oriented Projective Geometry*. Academic Press, San Diego, CA / London, UK, 1991.

[17] Marc Polleyfeys, *Self Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences.* Ph.D Thesis, Katholieke Universiteit Leuven. 1999.

[18] R. I. Hartley. Lines and points in three views – an integrated approach. *In Proceedings of the ARPA IU Workshop*. DARPA, 1994.

[19] Andrew Zisserman. *Geometric Framework for Vision I: Single View and Two-*View Geometry. 1998

[20] P.H.S. Torr. *Motion Segmentation and Outlier Detection*. Ph. D. Thesis, University of Oxford, 1995.

[21] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons, New York. 1987

[22] A. Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. *International Conference on Computer Vision*, 1995.

[23] P.H.S. Torr and A. Zisserman. Robust Parameterization and Computation of the Trifocal Tensor. P*roc. British Machine Vision Conference.* Pp 655-664. 1996.

[24] O. Faugeras and T. Papadopoulo. A nonlinear method for estimating the Projecttive geometry of 3 views. *Sixth International Conference on Computer Vision*, 1998 pp 477-484.

[25] M. E. Spetsakis and J. Aloimonos. Structure from Motion Using Line Correspondences. *The International Journal of Computer Vision*, 4:171–183, 1990.

[26] M. E. Spetsakis and .J. Aloimonos. A unified theory of structure from motion. *Proc. DARPA IU Workshop*, pages 271–283, 1990.

[27] B. Triggs. The geometry of projective reconstruction: Matching constraints and the joint image. *In Proc ICCV,* 1995

[28] R. I. Hartley. Computation of the Quadrifocal Tensor. *Computer Vision, ECCV'98,* Springer Verlag *1998* pp. 20-35.

[29] O.D. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between N images. *Proc. 5th International Conference on Computer Vision (ICCV 95),* Cambridge, MA, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp.951-956

[30] Theo Moons, A Guided Tour Through Multiview Relations. *In SMILE*. 1998, pp 304-346

[31] A. Shashua, “Algebraic functions for recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 8, pp. 779-789, 1995.

[32] R. Hartley, “A linear method for reconstruction from lines and points," in Proceedings of the International Conference on Computer Vision, pp. 882{887, Cambridge, Mass., June 1995.

[33] R. Koch, M. Pollefeys, and L. VanGool, “Multi view-point stereo from uncalibrated video sequences," in ECCV'98, pp. 55-71, 1998.

[34] M. Pollefeys, R. Koch, M. Vergauwen, and L. VanGool, \Automatic generation of 3d models from photographs," in Proceedings Virtual Systems and MultiMedia, 1998.

[35] A. Fitzgibbon and A. Zisserman, “Automatic camera recovery for closed or open image sequences,” 5th European Conference on Computer Vision, (Freiburg, Germany), pp. 311-326, Springer Verlag, June 1998.

[36] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong, A robust technique for matching two uncalibrated images through the recovery of the unkown epipolar geometry," Artificial Intelligence Journal, vol. 78, pp. 87{119, October 1995.

[37] P. J. Rousseeuw, “Least median of squares regression,” Journal of American

Statistical Association, vol. 79, pp. 871-880, Dec. 1984.

[38] R. C. Bolles and M. A. Fischler, “A ransac-based approach to model fitting and its application to finding cylinders in range data,” in Seventh International Joint Conference on Artificial Intelligence, (Vancouver, British Colombia, Canada), pp. 637-643, 1981.

[39] H. Sawhney, Y. Guo, J. Asmuth, and R. Kumar, “Multi-view 3d estimation and applications to match move,” in 1999 IEEE Workshop on MuliView Modelling and Analysis of Visual Scenes, pp. 21-28, 1999.

[40] P. McLaughlin, “Gauge invariance in projective 3d reconstruction,” in IEEE Workshop on Multi-View Modelling and Analysis of Visual Scenes, pp. 37-44, IEEE Computer Society, 1999.

[41] Photomodeler by EOS Systems Inc. http:/www.photomodeler.com/.

[42] P. Besl, “Active, optical range imaging sensors," Machine Vision and Applications, vol. 1, no. 1, pp. 127-152, 1988.

[43] S. Smith and J. Brady, “Susan - a new approach to low level image processing," International Journal of Computer Vision, pp. 45-78, May 1997.

[44] C. Harris and M. Stephens, “A combined corner and edge detector," in Proceedings of the 4th lvey Vision Conference, pp. 147{151, 1988.

[45] R. Hartley, “In defense of the 8 point algorithm,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, 1997.

[46] P. Torr and A. Zisserman, “Robust parameterization and computation of the trifocal tensor,” Image and Vision Computing, vol. 15, no. 591-605, 1997.

[47] P. Jasiobedski, “Fusing and guiding range measurements with colour video images,” in Proceedinsg International Conference on Recent Advances in 3-D Digital Imaging and Modelling, (Ottawa, Ontario), pp. 339-347, IEEE Computer Society Press, 1997.

[48] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. International Joint Conference on Artificial Intelligence, pages 674-679, 1981.

[49] Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.

[50] Jianbo Shi and Carlo Tomasi. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition, pages 593-600, 1994.

[51] Stan Birchfield. Derivation of Kanade-Lucas-Tomasi Tracking Equation. Unpublished, May 1996.

[52] Étienne Vincent and Robert Laganière, Matching Feature Points in Stereo Pairs: A Comparative Study of Some Matching Strategies, in Machine Graphics & Vision , vol. 10, no. 3, pp. 237-259, 2001

[53] R. Klette, K. Schluns, and A. Koschan, Computer Vision: three-dimensional data from images. Springer, 1996.

[54] A. Hampapur, R. Jain, and T. E. Weymouth. Production Model Based Digital Video Segmentation. *Multimedia Tools and Application*s, Vol.1, pp. 9-45, 1995.

[55] R. Lienhart. Reliable Transition Detection In Videos: A Survey and Practitioner's Guide. International Journal of Image and Graphics (IJIG), Vol. 1, No. 3,

pp. 469-486, 2001.

[56] U. Gargi, R. Kasturi, S. H. Strayer. Performance Characterization of Video-Shot-Change Detection Methods. *IEEE Transaction on Circuits and Systems for Video Technology*,Vol.10,No.1,Feb. 2000.

[57] G. Lupatini, C. Saraceno, and R. Leonardi. Scene Break Detection: A Comparison. *Research Issues in Data Engineering, Workshop on Continuous Media Databases and Application*s, pp. 34-41.1998.

[58] B. Shahraray. Scene Change Detection and Content-Based Sampling of Video Sequences. *SPIE Digital Video Compression, Algorithm and Technologie*s, Vol. 2419, pp. 2-13, 1995

[59] B.-L.Yeo and B. Liu. Rapid Scene Analysis on Compressed Video. *IEEE Transactions on Circuit and Systems for Video Technology*,Vol.5,No.6,Dec.1993.

[60] M. M. Yeung and B.-L. Yeo. Video Visualization for Compact Presentation and Fast Browsing of Pictorial Content. *IEEE Transactions on Circuits and Systems for Video Technology*,Vol.7, No. 5, pp. 771-785, Oct. 1997.

[61] J. Mateer, J. Robinson, Semi-Automated Logging for Professional Media Applications. Video, Vision and Graphics (VVG) 2003, Bath, UK, July, 2003.

[62] A Whitehead. Fast Feature Based Video Segmentation and Annotation. Proc. 7th International Symposium on Signal Processing and its Applications (ISSPA), Paris, 2003.

[63] S. Pfeiffer, R.Lienhart, G. Kühne, W. Effelsberg. The MoCA Project - Movie Content Analysis Research at the University of Mannheim. Informatik '98, pp. 329-338, 1998.

[64] J. Lee and B. Dickinson, "Multiresolution video indexing for subband coded video databases", in *Proceedings of IS&T/SPIE, Conference on Storage and Retrieval for Image and Video Databases,San Jose, CA*, 1994.

[65] R. Lienhart. Dynamic Video Summarization of Home Video. *SPIE Storage and Retrieval forMedia Dat bases 200*0, Vol. 3972, pp. 378-389, Jan. 2000.

[66] R. Lienhart, S. Pfeiffer, and W. Effelsberg. Video Abstracting. *Communications of the AC*M, Vol. 40, No. 12, pp. 55-62, Dec. 1997.

[67] A. Seyler, "Probability distribution of television frame difference", *Proc. Institute of Radio Electronic Engineers of Australia* 26(11), pp 355-366, 1965

[68] A. Nagasaka and Y. Tanaka, "Automatic video indexing and full-video search for object appearances", in *Visual Database Systems II,* pp 113-127, 1992

[69] H. Zhang, A. Kankanhalli, S. Smoliar, "Automatic partitioning of full-motion video", *ACM/Springer Multimedia Systems.* 1(1), pp 10-28,1993

[70] R Zabih, J. Miller, and K. Mai, "A Feature-Based Algorithm for Detecting and Classifying Scene Breaks", Proc. ACM Multimedia, pp. 189-200, 1995

[71] R Zabih, J. Miller, and K. Mai,. "A Feature Based Algorithm for detecting and Classifying Production Effects", *Multimedia Systems,* Vol 7, p 119-128, 1999.

[72] J. Canny A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 8, No. 6, Nov 1986.

[73] A Smeaton et al., "An Evaluation of Alternative Techniques for Automatic Detection of Shot Boundaries in Digital Video" in *Irish Machine Vision and Image Processing Conference*, 1999

[74] W. Hardle and D. Scott. “Smoothing in by weighted averaging using rounded points”, *Computational Statistics* Vol. 7: 97-128, 1992.

[75] R. Hartley, “Kruppa's equations derived from the fundamental matrix," IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 19, pp. 133-135, February 1997.

[76] Q.-T. Luong and O.D.Faugeras, “Self-calibration of a moving camera from point correspondences and fundamental matrices," International Journal of Computer Vision, vol. 22, no. 3, pp. 261-289, 1997.

[77] L. Lourakis and R. Deriche, “Camera self-calibration using the svd of the fundamental matrix," Tech. Rep. 3748, INRIA, Aug. 1999.

[78] P. Mendonca and R. Cipolla, “A simple technique for self-calibration," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, (Fort Collins, Colorado), pp. 112-116, June 1999.

[79] C. Zeller and O. Faugeras, “Camera self-calibration from video sequences: the kruppa equations revisited,” Tech. Rep. 2793, INRIA, Feb. 1996.

[80] M. Pollefeys and L. Van Gool, “Stratified Self-Calibration with the Modulus Constraint”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 21, No.8, pp.707-724, 1999.

[81] M. Pollefeys, R. Koch, and L. V. Gool, “Self-calibration and metric reconstruction in spite of varying and unknown intrinsic camera parameters," International Journal of Computer Vision, vol. 32, no. 1, pp. 7-25, 1999.

[82] W. Triggs. Autocalibration from planar scenes. In Proc. ECCV, 1998.

[83] P. Sturm and S. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In IEEE Conf. CVPR 1999.

[84] B. Triggs. Autocalibration from Planar Scenes. ECCV, pp. 89-105, 1998.

[85] Z. Zhang, “A flexible new technique for camera calibration”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330-1334, 2000.

[86] O. Faugeras and Q.T. Luong, The Geometry of Multiple Images. The MIT Press, 2001.

[87] P. Sturm, A case against kruppa’s equations for camera self-calibration,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pp. 1199-1204, Oct. 2000

[88] S. Bougnoux, “From projective to Euclidean space under any practical situation, a criticism of self-calibration,” in Proc. 6th Int. Conf. on Computer Vision, (Bombay, India), pp. 790-796, 1998.

[89] A. Morgan, Solving polynomial systems using continuation for science and engineering. Prentice Hall, Englewoord Clifis, 1987

[90] A. Fusiello, Uncalibrated Euclidean reconstruction: a review,” Image and Vision Computing, vol. 18, pp. 555-563, 2000.

[91] W. H. Press and B. P. Flannery, Numerical recipes in C. Cambridge University press, 1988.

[92] M. Maza and D. Yuret, “Dynamic hill climbing,” AI Expert, pp. 26{31, 1994.

[93] R. Hartley and A. Zisserman, Multiple view geometry in computer vision. Cambridge University Press, 2000.

[94] Nicholson, Linear Algebra with Applications (3rd. ed). PWS Publishing Com-

pany, 1995.

[95] J. Holland, “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1995

[96] K. A. De Jong, “An analysis of the behavior of a class of genetic adaptive systems,” Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1975.

[97] D. Laurendeau, G. Roth, and L. Borgeat, “Optimization algorithms for range image registration” Vision Interface 96, pages 141-151, Toronto, Canada.

[98] P. Torr, A.Zisserman, S Maybank, “Robust detection of Degenerate Configurations for the fundamental matrix” In Proc. 5th Int’l Conf. on Computer Vision, Boston, pages 1037—1042, 1995

[99] G. Roth and A. Whitehead, “Using projective vision to find camera positions in an image sequence,” in VI 2000, Montreal Canada, pp. 87-94f, May 2000.

[100] A. Whitehead and G. Roth, “The Projective Vision Toolkit”, in Proceedings Modeling and Simulation, (Pittsburgh, Pennsylvania), May 2000.

[101] T. Ueshiba and F. Tomita, “A factorization method for projective and Euclidean reconstruction," in ECCV'98, 5th European Conference on Computer Vision, (Freiburg, Germany), pp. 290-310, Springer Verlag, June 1998.

[102] ISPRS Working Group 2, Scene Modeling and VR. http://www.vit.iit.nrc.ca/elhakim/WGV2-data.html.

[103] P.Gurdjos and P.Sturm. “Methods and Geometry for Plane-Based Self-Calibration.” In Proc. of the International Conference on Computer Vision and Pattern Recognition, 2003

[104] H. Huang, C. Kao, Y. Lin, Y. Hung, Yi-Ping, “Disparity-based view interpolation for multiple-perspective stereoscopic displays”, *Proceedings of SPIE Vol. 3957, Stereoscopic Displays and Virtual Reality Systems VII*, p. 102-113, 2000

[105] Lily Lee, Raquel Romano, Gideon Stein, “Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame”, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Special Section on Video Surveillance and Monitoring, 22(8), 2000

[106] J. Kang, I. Cohen, G. Medioni. “Continuous multi-views tracking using tensor voting”, *Proceedings of Workshop on Motion and Video Computing, 2002.* pp 181- 186

[107] Y. Caspi and M. Irani. “Alignment of non-overlapping sequences”. *Proceedings of International Conference on Computer Vision*, Vancouver, BC, pp 76-83, 2001.

[108] S. Kuthirumal, C.V. Jawahar, and P.J. Narayanan. “Video frame alignment in multiple views”. *Proceedings of International Conference on Image Processing*, Rochester, NY, 2002.

[109] C. Rao, A.Gritai, M. Shah. “View-invariant Alignment and Matching of Video Sequences”, *In Proceedings of International Conference on Computer Vision*, pp 939-945, 2003.

[110] P. Tresadern and I. Reid. “Synchronizing Image Sequences of Non-Rigid Objects”, *In Proceedings of British Machine Vision Conference*, 2003

[111] David G. Lowe, “Distinctive image features from scale-invariant keypoints”, *International Journal of Computer Vision*, 2004. to appear

Index

A

aspect ratio, x, 21, 90, 93, 97, 101, 106, 107, 108, 110
autocalibrate, 20, 42, 90, 94, 96, 104, 106, 108, 109, 149
autocalibrated, 106, 107
autocalibrating, 106
Autocalibration, 90

B

blocksworld, 2
boundary detection, 5, 62

C

calibrated, 3, 9, 25, 27, 39, 42, 50, 94, 115
calibration, iii, 4, 7, 9, 20, 21, 27, 28, 38, 39, 40, 42, 43, 54, 57, 60, 61, 90, 91, 92, 93, 94, 97, 99, 101, 103, 104, 105, 110, 140, 149, 154
camera sequence, 121, 122, 123
candidate set, 73, 78, 84, 85, 89, 128, 129, 130, 131, 134
Class A, 91, 93, 105, 110
Class B, 90, 91, 92, 93, 104, 105, 110
Class C, 90, 91, 92
collineation, 12
computational video, ii, iii, 3, 4, 5, 6, 8, 9, 139
conic, 14, 16, 19, 93, 95, 96
correspondence, ii, iii, 4, 5, 25, 29, 32, 33, 39, 41, 43, 45, 50, 51, 54, 57, 60, 61, 65, 90, 115, 126, 127
cross camera subset, 113, 121
cut, viii, 5, 46, 47, 62, 63, 64, 65, 66, 70, 71, 72, 73, 74, 78, 80, 81, 82, 83, 85, 86, 87, 88, 89
cut detection, 5, 47, 62, 63, 65, 80, 82, 83, 87, 89

D

density, 67, 73, 74, 77, 78, 79, 83
disparity, viii, 5, 25, 26, 39, 49, 50, 51, 52, 58, 60, 61, 70, 137
dissolve, 62, 88
distribution, 29, 64, 82, 87, 100, 153
dynamic-CSS, 122

E

epipolar geometry, 25, 26, 27, 29, 30, 31, 51, 90, 95, 115, 149, 151
essential matrix, x, 9, 14, 27, 28, 51, 93, 94
exact synchronization, 114, 115, 116, 117, 132
extrinsic, 7, 20, 21, 22, 115

F

F1, 78, 80, 81, 82, 83, 84, 85, 86, 89
fade, 88
feature tracking, viii, 5, 34, 45, 62, 67, 68, 69, 72, 83, 85, 87, 89, 124, 125, 138, 140
focal length, x, 11, 58, 90, 93, 101, 105, 106, 107, 108, 109, 110, 111, 112, 141
frame rate, 4, 35, 43, 44, 55, 114, 115, 116, 117, 118, 119, 122, 125, 126, 130, 131, 132, 133, 138
frequency, 74, 116
full frame synchronization, 114, 115, 116, 117, 118, 126, 132, 133, 134, 136
fundamental matrix, v, x, 4, 6, 9, 28, 29, 30, 31, 33, 38, 41, 42, 51, 52, 53, 60, 61, 90, 91, 92, 93, 94, 95, 96, 97, 102, 103, 104, 106, 110, 111, 149, 154, 155

G

gradient, 34, 39, 44, 48, 50, 51, 52, 58, 60, 61, 68, 87, 93, 97, 100, 101, 102, 107, 124

H

homographies, 23
homography, x, 23, 90, 114

I

inflection point, 116, 123, 126, 127, 130, 131, 133, 134, 137
intrinsic, iii, x, 6, 16, 20, 21, 28, 90, 94, 102, 104, 105, 106, 108, 110, 112, 114, 141, 154
invariant, 15, 17, 19, 91, 108, 140, 155

M

minimum spanning tree, 70, 71, 72

O

optical center, x

P

PDF, viii, 78, 79
period, 116, 119, 137
phase shift, 120
photogrammetry, 4, 5, 57, 61, 113
planar, 2, 25, 52, 90, 114, 115, 140, 154
precision, 30, 78, 80, 81, 82, 84, 89, 115
primary synchronization period, 118

primary synchronization point, 116, 117, 118, 121, 122, 130
principle point, 104
probability density function, 73, 74, 75, 78

R

recall, 78, 80, 81, 82, 84, 89
reconstruction, iii, 6, 20, 39, 40, 42, 43, 45, 54, 57, 60, 61, 90, 92, 104, 105, 106, 110, 111, 149, 151, 152, 154, 155

S

salient frame, 43, 44, 45, 46
secondary synchronization point, 117, 118
segmentation, iii, 6, 62, 63, 89, 139, 141
skew, x, 90, 97, 104
static-CCS, 121, 122, 123, 124, 130
stereo vision, 3, 6, 25, 27, 42
support set, 51, 52, 60, 61

T

trajectory image, vi, viii, ix, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 134, 135, 137
trifocal tensor, v, 30, 31, 53, 60, 61, 122, 131, 149, 152
trilinear tensor, 4, 31, 38, 39, 40, 41, 42, 52, 53, 124, 150, 151

U

uncalibrated, 4, 9, 20, 27, 28, 38, 51, 60, 90, 108, 150, 151

V

video sequence, iii, viii, 4, 5, 6, 8, 43, 45, 54, 55, 61, 62, 63, 66, 68, 74, 76, 113, 114, 115, 116, 117, 118, 119, 120, 121, 124, 126, 127, 132, 139, 140, 151, 154

W

wipe, 62