ROBUST STATISTICS OVER RIEMANNIAN MANIFOLDS FOR COMPUTER VISION

BY RAGHAV SUBBARAO

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Peter Meer

and approved by

New Brunswick, New Jersey

May, 2008

ABSTRACT OF THE DISSERTATION

Robust Statistics over Riemannian Manifolds for Computer Vision

by Raghav Subbarao

Dissertation Director: Peter Meer

The nonlinear nature of many compute vision tasks involves analysis over curved non-linear spaces embedded in higher dimensional Euclidean spaces. Such spaces are known as manifolds and can be studied using the theory of differential geometry. In this thesis we develop two algorithms which can be applied over manifolds.

The *nonlinear mean shift* algorithm is a generalization of the original mean shift, a popular feature space analysis method for vector spaces. Nonlinear mean shift can be applied to any Riemannian manifold and is provably convergent to the local maxima of an appropriate kernel density. This algorithm is used for motion segmentation with different motion models and for the filtering of complex image data.

The *projection based M-estimator* is a robust regression algorithm which does not require a user supplied estimate of the scale, the level of noise corrupting the inliers. We build on the connections between kernel density estimation and robust M-estimators and develop data driven rules for scale estimation. The method can be generalized to handle heteroscedastic data and subspace estimation. The results of using pbM for affine motion estimation, fundamental matrix estimation and multibody factorization are presented.

A new sensor fusion method which can handle heteroscedastic data and incomplete estimates of parameters is also discussed. The method is used to combine image based pose estimates with inertial sensors.

Acknowledgements

Thanks to: Peter Meer for his mentorship, patience and advice. Oncel Tuzel for all the arguments, discussions and ideas. Yakup Genc for giving me an opportunity to work at Siemens Corporate Research. All my friends for making the graduate school experience a memorable one. My family for their support and understanding.

I would also like to thank Peter and Ioana Meer for all the dinners and movie recommendations. They made these last four years so much more enjoyable.

Dedication

To my parents

Table of Contents

Abstract ii

Acknowledgements iii

Dedication iv

List of Tables ix

List of Figures x

1. Introduction 1

Contributions of this thesis 5

2. Riemannian Geometry 8

2.1. Basic Algebraic Concepts 8

2.1.1. Groups 9

2.1.2. Topological Space 11

2.1.3. Homeomorphisms 11

2.1.4. Analytic Functions 12

2.2. Analytic Manifolds 13

2.2.1. Manifolds 13

2.2.2. Functions and Mappings 14

2.2.3. Tangent Spaces 15

2.3. Riemannian Manifolds 16

2.3.1. Riemannian Metric 17

2.3.2. Exponential and Logarithm Operators 18

2.3.3. Differential Operators on Manifolds 20

2.3.4. Covariant Derivatives 22

2.3.5. Parallel Transport . . . 23
2.3.6. Function Optimization . . . 23
2.4. Types of Riemannian Manifolds . . . 24
2.4.1. Lie Groups . . . 24
2.4.2. Homogeneous Spaces . . . 26
2.4.3. Grassmann Manifolds . . . 28
2.4.4. Essential Manifold . . . 29
2.4.5. Symmetric Positive Definite (SPD) Matrices . . . 34

3. Nonlinear Mean Shift . . . 36
3.1. Introduction . . . 36
3.2. The Original Mean Shift . . . 38
3.2.1. Mean Shift as Bounds Optimization . . . 38
3.3. Nonlinear Mean Shift . . . 40
3.3.1. Kernel Density Estimation over Riemannian Manifolds . . . 40
3.3.2. Mean Shift over Riemannian Manifolds . . . 41
3.4. Computational Details of Nonlinear Mean Shift . . . 42
3.5. Theoretical Properties . . . 44
3.5.1. Mean Shift and Expectation Maximization . . . 46
3.6. Applications and Results . . . 48
3.6.1. Motion Segmentation . . . 48
3D Translational Motion . . . 49
Affine Motion . . . 51
Camera Pose Segmentation . . . 52
Multibody Factorization . . . 53
Epipolar Segmentation . . . 55
3.6.2. Robust Estimation . . . 56
Mean Shift versus RANSAC . . . 57
Camera Tracking System . . . 59

3.6.3. Discontinuity Preserving Filtering 60
Chromatic Noise Filtering 61
DT-MRI Filtering 62
4. Projection Based M-Estimators 64
4.1. Previous Work 65
4.1.1. RANSAC and Robust Regression 66
4.1.2. Robust Subspace Estimation 67
4.1.3. Scale Independent Robust Regression 69
4.2. Robust Heteroscedastic Linear Regression 70
4.2.1. The Scale in RANSAC 72
4.2.2. Weighted Kernel Density Estimation 73
4.2.3. Reformulating the Robust Score 74
4.2.4. Projection Pursuit 77
4.3. Robust Subspace Estimation 78
4.4. The Projection Based M-estimator 80
4.4.1. Data-driven Scale Selection 80
4.4.2. Local Optimization 82
4.4.3. Inlier-Outlier Dichotomy Generation 83
4.5. Results 85
4.5.1. Synthetic Data 85
4.5.2. Affine Motion Estimation 87
4.5.3. Fundamental Matrix Estimation 89
Corridor Images 90
Merton College Images 92
Valbonne Images 93
4.5.4. Affine Factorization 95
Lab Sequence 96
Toy Car Sequence 98

4.6. Connection to Nonlinear Mean Shift 99

5. Fusion of Multiple Sources in Structure from Motion 100

5.1. Previous Work 101

5.2. Heteroscedastic Errors-in-Variables Algorithm 104

5.2.1. An Approach Through Linearization 105

5.2.2. The Heteroscedastic Errors-in-Variables Algorithm 107

5.2.3. Enforcing Ancillary Constraints 108

5.2.4. Camera Pose Estimation 109

5.2.5. Point Triangulation 110

5.3. Camera Tracking System 111

5.4. The Modified HEIV Algorithm 115

5.4.1. Relationship to Kalman Filtering Approaches 117

5.4.2. Derivation 118

5.4.3. Statistical Properties 123

5.5. Experimental Results 123

6. Conclusion and Future Work 131

6.1. Future Work 132

6.1.1. The Radon Transform 132

6.1.2. Other Manifolds in Vision 133

Appendix A. Conjugate Gradient over $\mathbf{G}_{N,k} \times \mathbb{R}^k$ 135

References 137

Vita 145

List of Tables

3.1. Formulae for *exp* and *log* operators over different manifolds 43

4.1. Kernel Density Estimation and M-Estimators 75

4.2. Performance Comparison - Affine Motion Estimation 89

4.3. Performance Comparison - *Corridor* Image Pair 91

4.4. Performance Comparison - *Merton College* Image Pair 93

4.5. Performance Comparison - *Valbonne* Image Pair 94

4.6. Segmentation Results of Factorization 97

4.7. Segmentation Results of Factorization 98

List of Figures

2.1. Example of a two-dimensional manifold. Two overlapping coordinate charts are shown. If the manifold is analytic, the transition map $\phi \circ \psi^{-1}$ (and $\psi \circ \phi^{-1}$) from $\mathbb{R}^2$ to $\mathbb{R}^2$ should be analytic. 14

2.2. Example of a two-dimensional manifold and the tangent space at the point $\mathbf{x}$. 19

2.3. Four different camera geometries which give the same essential matrix. In each row the geometries differ by changing the sign of the direction of translation. Each column is a twisted pair. The image was taken from [57, p.241]. 30

3.1. *3D Translational Motion. Mean shift over $G_{3,1}$.* In the left figure all the points are shown while on the right the inliers returned by the system. The table on the left contains the properties of the first four modes. Only the first three modes correspond to motions. The table on the right compares the results with the manual segmentations. 50

3.2. *Affine motion segmentation. Mean shift over $A(2)$.* In the left figure all the points are shown, and on the right only the inliers are shown. The table on the left contains the properties of the first four modes. Only the first three modes correspond to motions. The table on the right compares the results with the manual segmentations. 52

3.3. *Camera Pose Segmentation.* Images used to reconstruct 3D point clouds using the OpenCV implementation of the calibration technique of [142] 53

3.4. *Camera Pose Segmentation. Mean shift over* $\mathbf{SE}(3)$. The figure shows inliers for the different motions found. The table on the left contains the properties of the first four modes. Only the first three modes are valid motions. The table on the right compares the result with the manual segmentation. 54

3.5. *Multibody Factorization. Mean shift over* $G_{10,3}$. The left figure shows the first frame with all the points which are tracked. The middle and right images show the second and fifth frames with only the inliers. The table on the left contains the properties of the first four modes. Only the first three modes correspond to motions. The table on the right compares the results with the manual segmentations. 55

3.6. *Mean shift over the Essential Manifold.* The left figure shows the first frame with all the points which are matched. The right image shows the second frame with only the inliers returned by the segmentation. The table on the left contains the properties of the first three modes. Only the first two modes correspond to motions. The table on the right compares the results with the manual segmentations. 56

3.7. Comparison of the error densities for RANSAC and averaging as given by (3.28) and (3.29). (a) $n = 10$ for both curves. (b) $n = 100$ for both curves. (c) $n = 100$ for RANSAC and $n = 25$ for averaging. 58

3.8. Results of the camera tracking. The scene used is shown on the left. The reconstructed point cloud and camera frames are on the right. 60

3.9. *Chromatic Noise Filtering.* The *baboon* image corrupted with chromatic noise is shown on the left. The results of using standard mean shift filtering with EDISON are in the middle and the results of our method are on the right. . . 61

3.10. *Chromatic Noise Filtering.* The *jellybeans* image corrupted with chromatic noise is shown on the left. The results of using standard mean shift filtering with EDISON are in the middle and the results of our method are on the right. 62

3.11. Real DTI data of a human heart before and after smoothing. The jitter in the top image is due to noisy voxels having different anisotropies from their surroundings. These are removed by the smoothing and more continuous regions of uniform anisotropy are visible below. 63

4.1. Biweight loss function, $\rho(u)$ and the biweight M-kernel function, $\kappa(u)$. Image (a) on the left is the *biweight* loss function and (b) is the corresponding *M-kernel* function. 69

4.2. The quasi-linear variation of M-estimator scores with scale. The parameters $\boldsymbol{\theta}$ and α are set to their true values while the scale s is varied. The dashed vertical line indicates the true scale of the noise corrupting the inliers. 74

4.3. An example of projection pursuit. The 2D data points and the two directions, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are shown in the middle image. The kernel density estimate of the projections along $\boldsymbol{\theta}_1$ is shown on the left. There is a clear peak at the intercept. The projections along $\boldsymbol{\theta}_2$ give a more diffuse density, as seen in the right figure. 77

4.4. The pbM algorithm uses two different scales. Data-driven scales are used for computing a robust M-scores. A new scale is then used to separate the inlier from the outliers. 81

4.5. Scale selection experiment with synthetic data with $k = 1$. Figure (a) compares the various scale estimators' performance as the number of outliers increase. Figure (b) shows the mode estimate computed on the same data sets and (c) shows a zoomed-in version of (b). 85

4.6. Images used for affine motion estimation. All the 51 points (inliers and outliers) matched in the two views are shown. 87

4.7. Results of affine motion estimation. The 19 inliers returned by pbM are shown in the two images. 88

4.8. Fundamental matrix estimation for the *corridor* images. Frame 0 and frame 9 are shown along with all the 127 point matches (inliers and outliers). 90

4.9. Results of fundamental matrix estimation for the *corridor* images. The 66 inliers returned by pbM and epipolar lines of the 8 outliers misclassified as inliers are shown. The reason for the misclassifications is explained in the text. 91

4.10. Fundamental matrix estimation for the *Merton College* images. Frames 0 and 2 are shown along with all the 135 point matches (inliers and outliers). 92

4.11. Results of fundamental matrix estimation for the *Merton College* images. The 68 inliers returned by pbM and epipolar lines of the 6 outliers misclassified as inliers are shown. The reason for the misclassifications is explained in the text. 93

4.12. Fundamental matrix estimation for the *Valbonne* sequence. Both images are shown along with the 85 point matches (inliers and outliers). 94

4.13. Results of fundamental matrix estimation for the *Valbonne* sequence. The 45 inliers returned by pbM and epipolar lines of the 3 outliers misclassified as inliers are shown. 95

4.14. *Lab* sequence used for factorization. The three objects move independently and define different motion subspaces. The left image shows the first frame with all the points (inliers and outliers). The right shows the fifth frame with the points assigned to the three motions marked differently. The first motion $\mathbf{M_1}$ corresponds to the paper napkin, the second motion $\mathbf{M_2}$ to the car and $\mathbf{M}_3$ to the book. 97

4.15. *Toy car* sequence used for factorization. The three cars move independently and define different motion subspaces, but the yellow and black cars define subspaces very close to each other. The left image shows the first frame with all the points (inliers and outliers). The right image shows the fourth frame with the points assigned to the three cars marked differently. The first motion $\mathbf{M_1}$ corresponds to the blue car, $\mathbf{M_2}$ to the yellow car and $\mathbf{M_3}$ to the black car. . 98

5.1. Outline of the tracking system. 112

5.2. Camera paths for *workspace* scene. 113

5.3. Comparison of camera tracking at frame 623 for the *workspace* experiment. The HEIV algorithm is used to get the pose and render the wireframe on the left and the LM algorithm is used on the right. 114

5.4. *Wireless Inertia Cube 3* and the wireless receiver. 115

5.5. Camera paths for experiment 1. The results of the simple HEIV algorithm are compared with our fusion algorithm. The discontinuity in the position estimates when the markers come back into view at frame 520 is an indication of the drift suffered by the system. The fusion based system clearly suffers from lesser drift. 124

5.6. Comparison of camera tracking at frame 519 of *experiment one*. On the left the fusion algorithm is used to get the pose and render the wireframe while the HEIV algorithm is used on the right. 125

5.7. Comparison of camera tracking at frame 130 for the planar scene. The fusion algorithm is used to get the pose and render the wireframe on the left and the HEIV algorithm is used for the right image. 126

5.8. Camera paths results for experiment two. The simple HEIV algorithm breaks down and loses track due to the planarity of reconstructed features. The fusion algorithm presented has some jitter but is much more stable. The HEIV pose estimate keeps flipping between the true pose and its mirror image. 127

5.9. Comparison of camera tracking at frame 471 for experiment three. The fusion algorithm is used to get the pose and render the wireframe on the left and the HEIV algorithm is used on the right. 128

5.10. Camera paths results for experiment three. The fusion algorithm presented here clearly has a much more stable path and suffers from less drift. 129

Chapter 1
Introduction

Computer vision problems can be broadly classified into low-level, mid-level and high-level tasks. The output of each stage is passed onto the next stage for further analysis. Low-level tasks extract meaningful 'image features' from raw pixel data. These features are then processed in the mid-level stage to obtain information about the scene. Finally, at the high-level stage, the system tries to extract semantic information from the geometric information obtained at the mid-level stage.

A common feature across these different levels of vision tasks is the nonlinear nature of problems. Consequently, feature analysis and estimation in vision problems involve curved nonlinear spaces embedded in higher dimensional Euclidean spaces. These curved spaces exhibit a significant amounts of smoothness but they are not vector spaces. Notions such as sums and differences of points in these spaces are not well defined. Such smooth, curved surfaces are referred to as *manifolds*. Manifolds are said to be *Riemannian* if they are endowed with a well defined notion of the distance between two points lying on them. We are interested in practical applications where it is possible to give a manifold a Riemannian structure based on geometric considerations of invariance [92].

There exist many different examples of Riemannian manifolds which appear in frequently in vision. In low level applications it is known that the space of color signals forms a conical space with hyperbolic geometry [73, 74, 75]. Novel imaging methods, especially in medical applications, also lead to new types of image data with their own complex geometries [38, 28, 72, 132].

At the mid-level stage, manifolds are important for the analysis of the projection relations between 3D scene points and 2D image points. Depending on the nature of the camera, the type of object motions present in the scene and the type of available information, these exist many different types of constraint equations relating 3D scene points with their 2D projections. These equations are parameterized by pose of the camera and the relative motion between the camera and scene. In most cases the space of possible motion parameters forms a Riemannian manifold. Examples of such motion manifolds include 2D and 3D rotations, 2D and 3D rigid transformations, image affine motions, camera pose matrices and essential matrices.

In high-level applications, the analysis of shapes in 2D and 3D requires the working with shape spaces. The space of faces and cyclic motions such as human gait also form manifolds, although such manifolds are not very well understood. Use of histograms and probability distributions as descriptors can be handled formally within the framework of *information geometry* [2], which is the study of the differential geometry of the manifold of probability distributions.

Most of these manifolds are well known outside vision and have been studied in fields such as differential geometry and physics. However, the degeneracies of visual data sometimes lead to manifolds, such as the essential manifold, which are specific to computer vision and have not been analyzed previously

In the past, vision research has concentrated on developing techniques for filtering, estimation and analysis over Euclidean spaces. Extending these algorithms to handle Riemannian manifolds offers the advantages of theoretical consistency and practical utility. In this thesis we present two algorithms which account for the nonlinear nature of vision problems. The first is a generalization of the *mean shift* algorithm which was originally derived for Euclidean space. The second is the *projection based M-estimator* (pbM) which is a user independent robust regression algorithm.

Many vision tasks can be solved within the framework of feature space analysis. The features are regarded as being drawn from some unknown probability distribution and the required geometric information is encoded in the parameters of this distribution. For the vision system to be completely autonomous, the feature space analysis algorithm needs to satisfy some requirements.

- Firstly, making assumptions about the nature of the distribution in the feature space can lead to incorrect results. This is because standard theoretical models, such as the assumption of additive Gaussian noise or unimodality, are not valid. Consequently, the feature space analysis should be *nonparametric*, *i.e.*, it should make no assumption about the nature of the unknown distribution.

- Secondly, the data being processed is noisy and the algorithm should have a *tolerance* for these errors in the data.

- Finally, the data may include points which do not belong to the distribution being estimated. This can occur either due to gross errors at earlier stages or due to the presence of multiple geometric structures. Since the system is quasi-autonomous it has no knowledge of which data points belong to which distribution. To account for this the feature space analysis algorithm should be *robust*.

A popular nonparametric feature space analysis algorithm is the *mean shift* algorithm [16, 25][42, p.535]. It satisfies all the above properties of being nonparametric, showing a tolerance for noise and being robust. Since it was introduced to the computer vision community [25], mean shift has been applied to numerous applications, such as (non-rigid) tracking [8, 23, 26, 32, 55, 143], image segmentation [25], smoothing [25] and robust fusion [15, 24].

This standard form of mean shift assumes that the feature space in which all the data lies is a vector space. We extend the mean shift algorithm to the class of Riemannian manifolds. We present the results of using this nonlinear mean shift algorithm for motion segmentation with various motion models and for the filtering of complex image data.

Another class of robust algorithms used in computer vision are *robust regression* algorithms which perform regression in the presence of outlier data without affecting the quality of the parameter estimate. We discuss a robust regression algorithm, the *projection based M-estimator* (pbM) [111, 113], which is user-independent and exhibits superior performance to other state-of-the-art robust regression algorithms. This is done by exploiting the similarity between robust M-estimator cost functions and kernel density estimation in Euclidean space. These advantages are particularly obvious in the presence of large numbers of outliers and multiple structures. The pbM algorithm takes into account the geometry of the parameter space in a local optimization step. The parameter space forms a Riemannian manifold and the local optimization is adapted to work over this space.

The thesis is organized as follows. In Chapter 2, we give an introduction to the theory of Riemannian manifolds which is used in the rest of the thesis. We begin with the definition of basic algebraic concepts such as groups and topologies. These are used to give a formal definition of the term manifold. Next, Riemannian manifolds, which are manifolds endowed with a distance metric, are introduced. Concepts such as tangents, vector fields and covariant derivatives are defined. The notion of parallel transport, which is necessary for the extension of gradient-based function optimization methods to manifolds, is also introduced.

In Chapter 3 the *nonlinear mean shift algorithm* is derived. The original mean shift over Euclidean space, has been justified in many different ways. Generalizing each of these methods to manifolds leads to a different form of mean shift. However, in deriving the nonlinear mean shift algorithm, we have concentrated on proposing an algorithm which retains the properties which made the original mean shift so popular. Namely, ease of implementation and good convergence properties. It is shown that the nonlinear mean shift is provably convergent to a local maxima of an appropriately defined kernel density. The nonlinear mean shift algorithm is used to develop a *motion segmentation* method which is able to estimate the number of motions present in the data and their parameters in a single step. The given data set is allowed to contain outliers. The nonlinear mean shift algorithm is also used for the filtering of *chromatic noise* from images and for the smoothing of *diffusion tensor data*, an important new medical imaging representation.

In Chapter 4 the projection based M-estimator is presented. A major hurdle to the deployment of robust regression algorithms in practical applications has been the necessity for user supplied parameters which reflect some knowledge of the world. Here we concentrate on one such parameter, the level of additive noise corrupting the inliers which known as the *scale* of the noise. The performance of the system is sensitive to the accuracy of these scale estimates. It is also possible that the scale varies over time and it is necessary for the system to adapt itself to the varying levels of noise. For example, in a real-time image based reconstruction system the amount of noise corrupting the tracked corner features will change depending on how fast the camera is moving. The projection based M-estimator is a user independent robust regression algorithm which does not require user defined scale estimates but uses data driven rules to adapt itself to the particular data set. The pbM algorithm builds on the equivalence between the original mean shift algorithm and robust M-estimators to get estimates of the scale of inlier noise. The various scale estimates used by pbM are completely nonparametric and make minimal assumptions about the nature of noise. We present a general version of the pbM algorithm which is able to handle heteroscedastic data and is also extended to subspace estimation.

Some work on sensor fusion is presented in Chapter 5. Inertial sensors are increasingly being combined with image based reconstruction systems to obtain higher performance reconstruction and tracking. These fusion methods have to deal with two problems. Firstly, the inertial sensors often give incomplete information about the pose of the camera. Secondly, the heteroscedastic nature of the data has to be taken into account. We propose a variation of the HEIV algorithm for heteroscedastic regression which can combine the information from inertial sensors with the image based pose to obtain improved tracking performance. The results are tested using a gyroscope rigidly attached to the camera which returns estimates of the relative rotational motion of the camera between frames.

Conclusions and directions for future work are presented in Chapter 6.

Contributions of this thesis

The following are some of the main contributions of this thesis.

- Generalization of the original mean shift algorithm to Riemannian manifolds. The new algorithm is valid over any Riemannian manifold. (Section 3.3)

- Proof of convergence of the nonlinear mean shift algorithm and the conditions necessary for this. Most practical examples of Riemannian manifolds satisfy these conditions. (Section 3.5)
- The geometry of the essential manifold, which is the space of all essential matrices, is discussed. (Section 2.4.4)
- Application of the nonlinear mean shift algorithm for
  - Translation viewed by a calibrated camera (Section 3.6.1)
  - Affine image motion (Section 3.6.1)
  - Camera pose based segmentation (Section 3.6.1)
  - Multibody Factorization (Section 3.6.1)
  - Robust Camera pose estimation (Section 3.6.2)
  - Chromatic Noise filtering (Section 3.6.3)
  - Diffusion Tensor MRI filtering (Section 3.6.3)
- General form of projection based M-estimator to handle heteroscedastic data and subspace estimation problems. (Section 4.4)
- We propose a novel robust cost function which gives improved performance, especially in the presence of heteroscedastic data. (Section 4.2.3)
- We use gradient based optimization to find local maxima over the search space for regression and subspace estimation, which is a Riemannian manifold. (Appendix A)
- Develop a new nonparametric user-independent method for inlier-outlier separation based on residual errors. (Section 4.4.3)
- Application of the pbM algorithm for
  - Fundamental Matrix estimation (Section 4.5.3)
  - Affine motion estimation (Section A)

  - Multibody Factorization (Section 4.5.4)

- Sensor fusion with incomplete data for structure from motion. The algorithm is tested in a structure from motion system with a gyroscope rigidly attached to the camera. The gyroscope only returns estimates of the relative rotation of the camera between frames. (Chapter 5)

Chapter 2

Riemannian Geometry

In this chapter we introduce some basic theory of differential geometry. These concepts are used in the later chapters for the developing robust statistical techniques for Riemannian manifolds. A thorough introduction to differential geometry can be found in [9, 90].

At an intuitive level, manifolds can be though of as smooth, curved surfaces embedded in higher dimensional Euclidean spaces. Riemannian manifolds are manifolds endowed with a distance measure which allows us to measure how similar or dissimilar (close or distant) two points are.

We follow the notation of [90] in this chapter for representing manifolds and their associated algebraic structures.

$\mathcal{M}, \mathcal{N}$	manifolds	$\mathbf{p}, \mathbf{q}$	points
$\mathcal{U}, \mathcal{V}$	open sets	ϕ, ψ	mappings
f, g	real-valued functions	V, W	vector fields

Note, we represent the points on manifolds by small bold letters, e.g., $\mathbf{x}, \mathbf{y}$. In some of our examples, the manifold is a set of matrices. Although matrices are conventionally represented by capital bold letters, when we consider them to be points on a manifold, we denote them by small letters.

2.1 Basic Algebraic Concepts

In this section basic algebraic concepts such as groups and topologies are introduced. These concepts are used to formally define manifolds.

2.1.1 Groups

A set G endowed with a binary operation, $\cdot$, is called a group if it satisfies the following properties:

- *Closure*: $g \cdot h \in G$, for any $g, h \in G$
- *Associativity*: $g \cdot (h \cdot k) = (g \cdot h) \cdot k$ for any $g, h, k \in G$
- *Identity*: There exists an $e \in G$ such that $g \cdot e = e \cdot g = g$ for any $g \in G$. The element e is said to be the *identity* of the group.
- *Inverse*: For any $g \in G$, there exists a unique $h \in G$ such that $g \cdot h = h \cdot g = e$. The element h is said to be the *inverse* of g and denoted by g^{-1}.

It is important to remember that the group is defined by the set G and the binary operation $\cdot$. It is possible to define different binary operations on the same underlying set of elements to obtain different groups. However, we often denote a group only by the set of elements and the group operation is understood from the context.

The simplest example of a group is the set of integers, $\mathbb{Z}$, under addition. The identity is 0 and the inverse of any integer x is $-x$. If the binary operation is chosen to be multiplication, the set of integers is no longer a group. For example the multiplicative inverse of 2 is 1/2 which is not an integer.

Let $\mathbb{Q}$ denote the set of rational numbers. Then the set $\mathbb{Q} \setminus \{0\}$ forms a group under multiplication with identity 1. It is necessary to remove 0 since the multiplicative inverse of 0 is not defined as there is no x such that $0.x = 1$.

Although the group operation is associative, it is not necessarily commutative, *i.e.*, it is not necessary for the condition

$$g \cdot h = h \cdot g \tag{2.1}$$

to hold. All our previous examples were of commutative groups. An example of a non-commutative group is the set of invertible $n \times n$ square matrices. This is an important group known as the *general linear group*, $\mathbf{GL}(n)$ with the group operation being matrix multiplication. It can be verified that closure and associativity hold. The identity of $\mathbf{GL}(n)$ is the $n \times n$ identity matrix $\mathbf{e_n}$. However, matrix multiplication is *not* commutative and $\mathbf{GL}(n)$ is not a commutative group.

A subset H of G is called a *subgroup* if H satisfies all the properties of being a group under the same binary operation. Since H has to satisfy the four properties of the group, it must contain the identity and for any element $h \in H$ it is necessary that $h^{-1} \in H$. A necessary and sufficient condition for H to be a subgroup is that for any $h_1, h_2 \in H$, $h_1 h_2^{-1}$ lies in H. For example the set of positive rational numbers, $\mathbb{Q}^+$ forms a subgroup of $\mathbb{Q}$ under multiplication. However, the set of negative rational numbers, $\mathbb{Q}^-$ is not a subgroup since $\mathbb{Q}^-$ does not contain the identity 1.

A more relevant example of a subgroup is the *special linear group*, $\mathbf{SL}(n)$ which is the set of $n \times n$ matrices with unit determinant

$$\mathbf{SL}(n) = \{\mathbf{X} \in \mathbb{R}^{n \times n} | det(\mathbf{X}) = 1\}. \tag{2.2}$$

Since the matrices in $\mathbf{SL}(n)$ have a determinant of one, they are invertible and they lie in $\mathbf{GL}(n)$. The identity $\mathbf{e_n}$ lies in $\mathbf{SL}(n)$ and the inverse of a matrix with unit determinant also has a determinant of one. Therefore, for any $\mathbf{X} \in \mathbf{SL}(n)$, $\mathbf{X}^{-1}$ also lies in $\mathbf{SL}(n)$.

Let H be a subgroup of the group G and let g be an element of G. The set $\{gh | h \in H\}$ is known as a *left coset* of H and is denoted by gH. We can similarly define *right cosets*, Hg. A basic results of group theory shows that the group G can be divided into disjoint cosets of H. Suppose $g_1 H$ and $g_2 H$ share a common member g. Then there exist h_1 and h_2 such that $g_1 h_1 = g_2 h_2 = g$. Now consider any element $g_1 h$ of the coset $g_1 H$. We have

$$g_1 h = g_1 h_1 (h_1^{-1} h) = g_2 h_2 (h_1^{-1} h) = g_2 (h_2 h_1^{-1} h). \tag{2.3}$$

Since H is a subgroup and $h, h_1, h_2 \in H$, $h_2 h_1^{-1} h \in H$ and $g_1 h = g_2 (h_2 h_1^{-1} h) \in g_2 H$. Therefore, $g_1 H \subseteq g_2 H$. We can reverse the process to show $g_2 H \subseteq g_1 H$. Therefore, the two sets are equal. The *factor space* G/H is the space of all *left cosets.* Each point in the factor space corresponds to a single coset.

2.1.2 Topological Space

Consider a set $\mathbf{X}$ and let $\mathcal{T}$ be a family of subsets of $\mathbf{X}$. The pair $(\mathbf{X}, \mathcal{T})$ forms a *topological space* if the following conditions hold.

- $\emptyset \in \mathcal{T}$ and $\mathbf{X} \in \mathcal{T}$, where $\emptyset$ is the empty set.
- The union of any family of sets in $\mathcal{T}$ also lies in $\mathcal{T}$
- The intersection of any finite number of sets in $\mathcal{T}$ belongs to $\mathcal{T}$.

The family $\mathcal{T}$ is said to be a *topology* of $\mathbf{X}$ and the sets in $\mathcal{T}$ are called the *open sets* of the topological space. A topological space is defined by the underlying set $\mathbf{X}$ and the topology $\mathcal{T}$ and different topologies can be defined on the same $\mathbf{X}$ to obtain different topological spaces. However, when the topology is clear from the context, a topological space is denoted simply by the underlying set $\mathbf{X}$. For a point $x \in \mathbf{X}$, any open set $\mathcal{U} \in \mathcal{T}$ which contains x is said to be a *neighbourhood* of x.

A topological space $\mathbf{X}$ is said to be *separated* or *Hausdorff* if for any two points $x, y \in \mathbf{X}$, there exist neighbourhoods $\mathcal{U}$ and $\mathcal{V}$ of x and y respectively, such that $\mathcal{U} \cap \mathcal{V} = \emptyset$, *i.e.*, any two points x and y can be separated by disjoint neighbourhoods.

The simplest example of a topology is the real line, $\mathbb{R}$. We start be defining all *open intervals*, $(a, b) = \{x \in \mathbb{R} | a < x < b\}$ to be open sets. We then add all the other sets necessary to satisfy the axioms listed above which are required to make this a topology. For example, sets of the form $(a, b) \cup (c, d)$ must be an open sets since they are the unions of open sets. Adding all such sets we get the *usual topology* on $\mathbb{R}$.

This idea can be extended to higher dimensional Euclidean spaces. For example the real plane, $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$ is the product of the real lines. Open sets in $\mathbb{R}^2$ are of the form $\mathcal{U} \times \mathcal{V} = \{(x, y) | x \in \mathcal{U}, y \in \mathcal{V}\}$ where $\mathcal{U}$ and $\mathcal{V}$ are open sets in $\mathbb{R}$.

2.1.3 Homeomorphisms

We will eventually be dealing with real valued functions defined over topological spaces. For functions on the real line we can define notions such as continuity and differentiability. It is not clear how to define continuity for functions over an abstract topological space. For functions defined on a real line continuity is defined by saying that as we move towards a point x, the value of the function gets closer to the value $f(x)$. Over a topological space, the idea of being 'close' to a particular point is captured by its neighbourhoods and the continuity of a function is defined by how it maps open sets of the topology. Given two topological spaces $\mathbf{X}$ and $\mathbf{Y}$, a mapping $f : \mathbf{X} \rightarrow \mathbf{Y}$ is *continuous* if for any open set $\mathcal{V} \in \mathbf{Y}$ the set

$$f^{-1}(V) = \{x \in \mathbf{X} | f(x) \in V\} \tag{2.4}$$

is an open set in $\mathbf{X}$. The set $f^{-1}(V)$ is known as the *pull-back* of V. A mapping f is called a *homeomorphism* if it is bijective (one-to-one and onto) and $f : \mathbf{X} \rightarrow \mathbf{Y}$ and $f^{-1} : \mathbf{Y} \rightarrow \mathbf{X}$ are both continuous.

2.1.4 Analytic Functions

A real valued function f from an open set $\mathcal{U} \subseteq \mathbb{R}^n$ to $\mathbb{R}$ is *smooth* if all its partial derivatives with respect to each coordinate and of any order exist at every point in $\mathcal{U}$. A smooth function for which these derivatives can be arranged into a convergent Taylor series is said to be *analytic*.

The *natural coordinate functions*, $u^i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \ldots, n$ are the functions which map each point in $\mathbb{R}^n$ to its coordinates. For $\mathbf{x} = (x_1, \ldots, x_n)$

$$u^i(\mathbf{x}) = x_i. \tag{2.5}$$

The coordinate functions are trivially smooth and analytic. A function ϕ from $\mathbb{R}^n$ to $\mathbb{R}^m$ is *smooth* provided each of its coordinate functions $u^i \circ \phi : \mathbb{R}^n \rightarrow \mathbb{R}$ is smooth. Similarly, a function ϕ from $\mathbb{R}^n$ to $\mathbb{R}^m$ is *analytic* provided each of its coordinate functions $u^i \circ \phi : \mathbb{R}^n \rightarrow \mathbb{R}$ is analytic.

The use of superscripts and subscripts for indexing components follows the standard rules based on whether they are covariant or contravariant tensors. However, we do not discuss this matter any further here.

2.2 Analytic Manifolds

The simplest example of a manifold is the usual Euclidean n-space $\mathbb{R}^n$. We are familiar with this space and take a lot of notions for granted such as the Euclidean metric and its connection with the natural inner product of vectors in $\mathbb{R}^n$. Analytical manifolds offer a formal algebraic structure to extend these properties to other spaces.

2.2.1 Manifolds

A manifold is a space that is locally similar to Euclidean space. This is achieved formally by building mappings which make each small patch of a manifold similar to an open set in Euclidean space. To do this, we need to answer the following questions. How do we find these patches which look similar to Euclidean space? How are the different patches related to each other?

Consider a topological space $\mathcal{M}$. A *coordinate chart* of a topological space is a neighborhood $\mathcal{U} \subseteq \mathcal{M}$ and an associated homeomorphism ϕ from $\mathcal{U}$ to some Euclidean space $\mathbb{R}^m$. By definition, $\phi(\mathcal{U})$ is an open set in $\mathbb{R}^m$. This mapping is written as

$$\phi(\mathbf{x}) = [\phi^1(\mathbf{x}), \ldots, \phi^m(\mathbf{x})] \tag{2.6}$$

for each $\mathbf{x} \in \mathcal{U}$ and the resulting functions $\phi^i, i = 1, \ldots, n$ are the *coordinate functions* of ϕ. We have the identity $u^i \circ \phi = \phi^i$, where u^i are the natural coordinate functions defined previously. The neighborhood $\mathcal{U}$ and its associated mapping ϕ together form a coordinate chart. The value m is the *dimension* of ϕ.

Consider two different m-dimensional coordinate charts $(\mathcal{U}, \phi)$ and $(\mathcal{V}, \psi)$ such that $\mathcal{U} \cap \mathcal{V}$ is nonempty. The *transition map* $\phi \circ \psi^{-1}$ is a mapping from the open set $\psi(\mathcal{U} \cap \mathcal{V}) \in \mathbb{R}^m$ to the open set $\phi(\mathcal{U} \cap \mathcal{V}) \in \mathbb{R}^m$. These ideas are graphically illustrated in Figure 2.1. The open sets $\mathcal{U}$ and $\mathcal{V}$ overlap. The image of the overlap region under the mapping ψ is shaded light grey and the image under ϕ is shaded dark gray. The transition map $\psi^{-1} \circ \phi$ maps points from the light grey region to the dark grey region. If the transition maps are smooth in the Euclidean sense defined above, then $\mathcal{U}$ and $\mathcal{V}$ are said to *overlap smoothly*. Coordinate charts are said to *compatible* if they do not overlap or overlap smoothly.

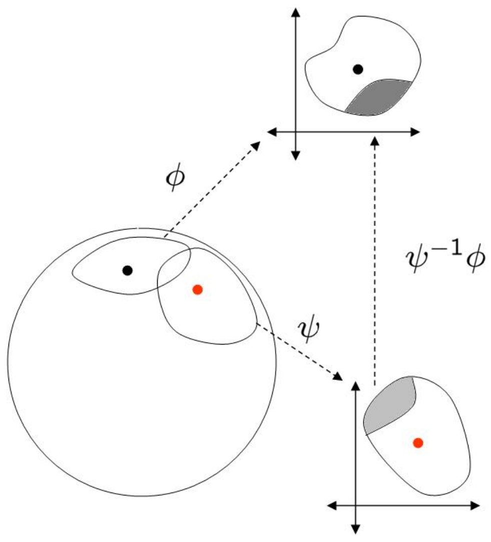


Figure 2.1: Example of a two-dimensional manifold. Two overlapping coordinate charts are shown. If the manifold is analytic, the transition map $\phi \circ \psi^{-1}$ (and $\psi \circ \phi^{-1}$) from $\mathbb{R}^2$ to $\mathbb{R}^2$ should be analytic.

An *atlas*, $\mathcal{A}$, of dimension m is a set of m-dimensional coordinate charts such that

- Each point of $\mathcal{M}$ is contained in some coordinate chart.
- Any two coordinate charts are compatible.

The value of m is said to be the *dimension* of the atlas. An atlas is said to be *complete* if it contains all the coordinate charts which are compatible with the coordinate charts of the atlas.

An *analytic manifold* is a Hausdorff, topological space furnished with a complete atlas. From now on we restrict ourselves to analytic manifolds.

2.2.2 Functions and Mappings

Consider a real valued function $f : \mathcal{M} \to \mathbb{R}$ on the manifold. Given a coordinate chart $(\mathcal{U}, \phi)$, the *coordinate expression* of f is the function $f \circ \phi^{-1}$ which maps the open set $\phi(\mathcal{U}) \in \mathbb{R}^m$ to $\mathbb{R}$. The function f is said to be *continuous*, if, for all coordinate charts, the coordinate expression is continuous in the Euclidean sense. Similarly, f is said to be *analytic* if the coordinate expression is analytic for all coordinate charts. If

$f, g : \mathcal{M} \to \mathbb{R}$ are analytic, then fg and $f + g$ are also analytic. The space of all real valued analytic functions over $\mathcal{M}$ is denoted by $\mathfrak{F}(\mathcal{M})$.

The idea of smooth mappings can be extended to mappings between manifolds. If η is a mapping from a manifold $\mathcal{M}$ to $\mathcal{N}$. Then η is smooth (analytic) if, for every coordinate chart $\phi : \mathcal{M} \to \mathbb{R}^m$ and $\psi : \mathcal{N} \to \mathbb{R}^n$, the mapping $\psi \circ \eta \circ \phi^{-1} : \mathbb{R}^m \to \mathbb{R}^n$ is smooth (analytic).

2.2.3 Tangent Spaces

The tangent space can be thought of as the set of allowed velocities for a point constrained to move on the manifold. Mathematically, it is a generalization of the idea of a directional derivative in Euclidean space. A tangent of $\mathcal{M}$ at $\mathbf{x}$ is a real-valued operator on continuous functions satisfying

$$\Delta(af + bh) = a\Delta(f) + b\Delta(h) \tag{2.7}$$

$$\Delta(fh) = f\Delta(h) + h\Delta(f) \tag{2.8}$$

for all continuous functions f, h and $a, b \in \mathbb{R}$. The set of all tangents at $\mathbf{x}$ is denoted by $T_{\mathbf{X}}(\mathcal{M})$. The real number assigned to f can be thought of as the derivative of f in the direction represented by Δ. Properties (2.7) and (2.8) ensure that the mapping is linear and satisfies the Leibniz product rule of derivatives.

The usual definitions of addition and scalar multiplication

$$(\Delta + \Gamma)(f) = \Delta(f) + \Gamma(f) \tag{2.9}$$

$$(a\Delta)(f) = a\Delta(f) \tag{2.10}$$

makes $T_{\mathbf{X}}(\mathcal{M})$ a vector space over $\mathbb{R}$. For m-dimensional manifolds, the tangent space is a m-dimensional vector space [90, Ch.1].

The value of any tangent acting on a particular function only depends on the behaviour of the function in a neighbourhood of the point. If there exists a neighbourhood $\mathcal{U}$ of $\mathbf{x}$ such that $f(\mathbf{y}) = g(\mathbf{y})$ for all $\mathbf{y} \in \mathcal{U}$, then for any tangent $\Delta \in T_{\mathcal{M}}(\mathbf{x})$, $(\Delta f)|_{\mathbf{X}} = (\Delta g)|_{\mathbf{X}}$.

When using the coordinate chart $(\mathcal{U}, \phi)$, a convenient basis for $T_{\mathbf{X}}(\mathcal{M})$ is the set of tangents $\partial_i|_{\mathbf{X}}, i = 1, \ldots, m$

$$\partial_i|_{\mathbf{X}}(f) = \left.\frac{\partial(f \circ \phi^{-1})}{\partial u^i}\right|_{\phi(\mathbf{X})} \tag{2.11}$$

where, u^i is the i-th coordinate function in $\mathbb{R}^m$. That is, ∂_i maps f to the directional derivative of $\tilde{f} = f \circ \phi^{-1}$ along the i-th coordinate, computed at $\phi(\mathbf{x}) \in \mathbb{R}^m$. Simple algebra shows that these operators satisfy the properties (2.7) and (2.8) and therefore lie in the tangent space $T_{\mathbf{X}}(\mathcal{M})$.

A *vector field*, V on a manifold is a function which assigns to each point $\mathbf{x}$ a tangent $V_{\mathbf{X}}$ in $T_{\mathbf{X}}(\mathcal{M})$. If V is a vector field and f is any continuous function on the manifold, Vf denotes the real valued function on the manifold such that

$$(Vf)(\mathbf{x}) = V_{\mathbf{X}}(f). \tag{2.12}$$

The vector field V is said to be smooth if the function Vf is smooth for all $f \in \mathfrak{F}(\mathcal{M})$. The set of all smooth vector fields is denoted by $\mathfrak{X}(\mathcal{M})$. For a coordinate chart $(\mathcal{U}, \phi)$, the vector field assigning the tangent $\partial_i|_{\mathbf{X}}$ to the point $\mathbf{x}$ is called the i-th *coordinate vector field*. The coordinate vector fields are smooth and any vector field can be expressed as a linear combination of these vector fields.

For two vector fields $V, W \in \mathfrak{X}(\mathcal{M})$, define the *bracket* as

$$[V, W] = VW - WV. \tag{2.13}$$

Therefore, $[V, W]$ is the vector field which sends a function f to the real valued function $V(Wf) - W(Vf)$.

2.3 Riemannian Manifolds

A *Riemannian manifold* is an ordered pair $(\mathcal{M}, g)$ consisting of the manifold and an associated metric. It is possible to define different metrics on the same manifold to obtain different Riemannian manifolds. However, in practice there exists a standard metric and the Riemannian manifold is denoted by the underlying manifold $\mathcal{M}$. This metric is chosen by requiring it to be invariant to some class of geometric transformations [92].

2.3.1 Riemannian Metric

A *bilinear form* on a vector space T is a mapping b from $T \times T$ to $\mathbb{R}$ which is linear in both parameters. Therefore, for any $u, v, w \in T$ and $\alpha, \beta \in \mathbb{R}$

- $b(\alpha u + \beta v, w) = \alpha b(u, w) + \beta b(v, w)$
- $b(w, \alpha u + \beta v) = \alpha b(w, u) + \beta b(w, v)$.

A *symmetric bilinear form* also satisfies $b(u, v) = b(v, u)$. A bilinear form is positive definite if $b(u, u) \geq 0$ with equality occurring if and only if $u = 0$. The dot product on any Euclidean space is an example of a symmetric positive definite bilinear form.

Riemannian geometry adds further structure to an analytic manifold by defining an symmetric, positive definite bilinear form $g_{\mathbf{X}}(\cdot, \cdot)$ on the tangent space at each point $\mathbf{x}$ on the manifold. This set of bilinear forms is known as the *metric tensor* and acts as a function which assigns to each point some bilinear form on its tangent space. To maintain continuity we require that if V, W are two smooth vector fields then the function $f(\mathbf{x}) = g_{\mathbf{X}}(V_{\mathbf{X}}, W_{\mathbf{X}})$ is a smooth function on the manifold. This is sufficient to ensure that the bilinear form varies smoothly as we move between points on the manifold.

For computational purposes, we can compute an explicit expression for the metric tensor. Given a coordinate chart $(\mathcal{U}, \phi)$, take the basis of the tangent space to be the coordinate basis $\partial_i, i = 1, \ldots, m$ defined earlier. In this basis the metric tensor can be expressed as a $m \times m$ matrix whose ij-th value is given by

$$g_{ij} = g(\partial_i, \partial_j). \tag{2.14}$$

Any tangent can be expressed as a linear combination of the basis vectors as $\Delta = \sum_i \Delta^i \partial_i$. The inner product of two vectors, Δ and Γ is given by

$$g(\Delta, \Gamma) = \sum_{i,j=1}^{m} g_{ij} \Delta^i \Gamma^j. \tag{2.15}$$

Choosing a different basis would lead to different coordinates for each vector and for the metric tensor. However, since the inner product is an inherent geometric notion that value of the inner product of Δ and Γ would be the same.

A *curve* is a continuous mapping α from the interval $[0, 1]$ to $\mathcal{M}$. For a particular $t \in [0, 1]$, $\alpha(t)$ lies on the manifold and $\alpha'(t)$ is the tangent at $\alpha(t)$ which maps a function f to $\partial(f \circ \alpha)/\partial t$. It can be verified the $\alpha'(t)$ satisfies the conditions (2.7) and (2.8) and lies in the tangent space at $\alpha(t)$. Physically $\alpha'(t)$ is the tangent which points along the curve $\alpha(t)$. The norm of $\alpha'(t)$ gives the speed of the curve. The length of the curve is given by

$$\int_{t=0}^{1} \sqrt{g_{\alpha(t)}(\alpha'(t), \alpha'(t))}dt. \tag{2.16}$$

Given two points $\mathbf{x}$ and $\mathbf{y}$ on the manifold, there will exist an infinite number of curves from $\mathbf{x}$ to $\mathbf{y}$, *i.e.*, with $\alpha(0) = \mathbf{x}$ and $\alpha(1) = \mathbf{y}$. The shortest such path from $\mathbf{x}$ to $\mathbf{y}$ is the *geodesic* from $\mathbf{x}$ to $\mathbf{y}$. The length of the geodesic is defined to be the *Riemannian distance* between the two points. Geodesics have the property that $g_{\alpha(t)}(\alpha'(t), \alpha'(t))$ is constant for all $t \in [0, 1]$, *i.e.*, the velocity is constant along the geodesic [90, Ch.3]. This property of having zero acceleration is sometimes used to define a geodesic.

2.3.2 Exponential and Logarithm Operators

For Riemannian manifolds, tangents in the tangent space and geodesics on the manifold are closely related. For each tangent $\Delta \in T_{\mathbf{X}}(\mathcal{M})$, there is a unique geodesic $\alpha : [0, 1] \to \mathcal{M}$ starting at $\mathbf{x}$ with initial velocity $\alpha'(0) = \Delta$. The *exponential map*, $exp_{\mathbf{X}}$, maps Δ to the point on the manifold reached by this geodesic

$$exp_{\mathbf{X}}(\Delta) = \alpha(1). \tag{2.17}$$

The origin of the tangent space is mapped to the point itself, $exp_{\mathbf{X}}(0) = \mathbf{x}$. For each point $\mathbf{x} \in \mathcal{M}$, there exists a neighborhood $\tilde{\mathcal{U}}$ of the origin in $T_{\mathbf{X}}(\mathcal{M})$, such that $exp_{\mathbf{X}}$ is a diffeomorphism from $\tilde{\mathcal{U}}$ onto a neighborhood $\mathcal{U}$ of $\mathbf{x}$ [90, Ch.3]. Over this neighborhood $\mathcal{U}$, we can define the inverse of the exponential and this mapping from $\mathcal{U}$ to $\tilde{\mathcal{U}}$ is known as the *logarithm map*, $log_{\mathbf{X}} = exp_{\mathbf{X}}^{-1}$. Note that the exponential and logarithm operators vary as the point $\mathbf{x}$ moves. This is made explicit by the subscript in the exponential and logarithm operators. The above concepts are illustrated in Figure 2.2, where $\mathbf{x}$, $\mathbf{y}$ are points on the manifold and $\Delta \in T_{\mathbf{X}}(\mathcal{M})$. The dotted line shows the geodesic starting

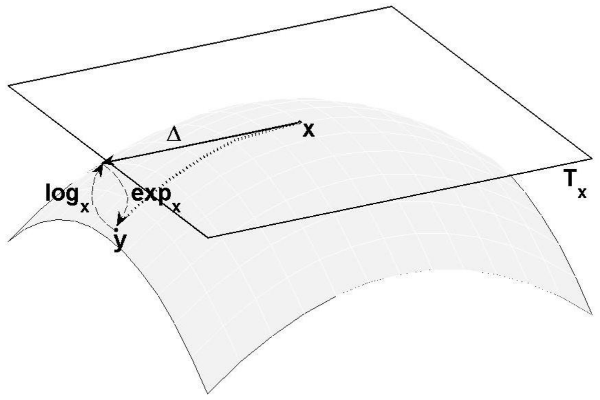


Figure 2.2: Example of a two-dimensional manifold and the tangent space at the point $\mathbf{x}$.

at $\mathbf{x}$ and ending at $\mathbf{y}$. This geodesic has an initial velocity Δ and we have $exp_{\mathbf{X}}(\Delta) = \mathbf{y}$ and $log_{\mathbf{X}}(\mathbf{y}) = \Delta$. The specific forms of these operators depend on the manifold. We present explicit formulae for certain manifolds in later sections.

The neighborhood $\tilde{\mathcal{U}}$ defined above is not necessarily convex. However, $\tilde{\mathcal{U}}$ is *star-shaped*, *i.e.*, for any point lying in $\tilde{\mathcal{U}}$, the line joining the point to the origin is contained in $\tilde{\mathcal{U}}$ [90, Ch.3]. The image of a star-shaped neighborhood under the exponential mapping is a neighborhood of $\mathbf{x}$ on the manifold. This neighborhood is known as a *normal neighborhood*.

The radius of the largest open ball in $T_{\mathbf{X}}(\mathcal{M})$, centered at the origin over which $exp_{\mathbf{X}}$ is invertible, is known as the *injectivity radius* at $\mathbf{x}$ and denoted by $i(\mathbf{x}, \mathcal{M})$. The injectivity radius of the manifold, $i(\mathcal{M})$, is the minimum of the injectivity radii at all points on the manifold $\mathcal{M}$

$$i(\mathcal{M}) = \min_{\mathbf{X} \in \mathcal{M}} i(\mathbf{x}, \mathcal{M}). \tag{2.18}$$

For any open ball centered at the origin in $T_{\mathbf{X}}(\mathcal{M})$ with a radius less than $i(\mathcal{M})$, the exponential map is one-to-one and its inverse is given by the logarithm.

The exponential map can be used to define convenient coordinates for normal neighborhoods, which simplify computation. Let $\tilde{\mathcal{U}}$ be a star shaped neighborhood at the origin in $T_{\mathbf{X}}(\mathcal{M})$ and let $\mathcal{U}$ be its image under the exponential map, *i.e.*, $\mathcal{U}$ is a normal neighborhood of $\mathbf{x}$. Let, $\mathbf{e}_i, i = 1, \ldots, m$ be any orthonormal coordinate system for $T_{\mathbf{X}}(\mathcal{M})$. Therefore,

$$g(\mathbf{e}_i, \mathbf{e}_j) = \begin{cases} 0 & if \;\; i \neq j \\ 1 & if \;\; i = j. \end{cases} \tag{2.19}$$

The *normal coordinate system* is the coordinate chart $(\mathcal{U}, \phi)$ which maps $\mathbf{y} \in \mathcal{U}$ to the coordinates of $log_{\mathbf{X}}(\mathbf{y})$ in the orthonormal coordinate system

$$log_{\mathbf{X}}(\mathbf{y}) = \sum_{i=1}^{m} \phi^i(\mathbf{y}) \mathbf{e}_i \tag{2.20}$$

where, $\phi^i(\mathbf{y})$ is the i-th coordinate of $\phi(\mathbf{y}) \in \mathbb{R}^m$ [90, Ch.3]. If we use (2.11) to define a define a basis of $T_{\mathbf{X}}(\mathcal{M})$ based on normal coordinates, we get $\partial_i = \mathbf{e}_i$.

2.3.3 Differential Operators on Manifolds

For a smooth, real valued function $f : \mathcal{M} \rightarrow \mathbb{R}$, the *gradient* of f at $\mathbf{x}$, $\nabla f \in T_{\mathbf{X}}(\mathcal{M})$, is the *unique* tangent vector satisfying

$$g_{\mathbf{X}}(\nabla f, \Delta) = \Delta f \tag{2.21}$$

for any $\Delta \in T_{\mathbf{X}}(\mathcal{M})$. The gradient is the unique tangent such that the directional derivative along any other tangent Δ is equal to the inner product of Δ with the gradient. It is possible to generalize higher order operators such as the Hessian and Laplacian for functions on manifolds [90, Ch.3]. Using these operators the usual function optimization techniques such as gradient ascent, Newton iterations and conjugate gradient can be generalized to manifolds [31, 106].

For two points, $\mathbf{x}, \mathbf{y} \in \mathcal{M}$ let $d(\mathbf{x}, \mathbf{y})$ be the Riemannian distance between them. Consider the function $f(\mathbf{x}) = d^2(\mathbf{x}, \mathbf{y})$ as a function of $\mathbf{x}$ measuring the squared distance from $\mathbf{y}$. This is a real function on the manifold and we have the following property,

Theorem 1. *The gradient of the Riemann squared distance is given by*

$$\nabla f(\mathbf{x}) = \nabla_{\mathbf{X}} \, d^2(\mathbf{x}, \mathbf{y}) = -2 \, log_{\mathbf{X}}(\mathbf{y}) \; . \tag{2.22}$$

This property is well known, for example [7, 36].

Proof 1. Let $\alpha : [0,1] \rightarrow \mathcal{M}$ be the geodesic from $\mathbf{x}$ to $\mathbf{y}$. Then, the geodesic from $\mathbf{y}$ to $\mathbf{x}$ is given by $\beta(t) = \alpha(1-t)$. For any $t \in [0,1]$, we have

$$\beta'(t) = -\alpha'(1-t). \tag{2.23}$$

It states that, for any point on the geodesic between $\mathbf{x}$ and $\mathbf{y}$, the tangents along the curves $\alpha(t)$ and $\beta(t)$ differ by multiplication by -1. Specifically, at $t = 1$ we get $\beta'(1) = -\alpha'(0)$. Recall that $\alpha'(0)$ and $log_{\mathbf{X}}(\mathbf{y})$ are two different expressions for the same initial velocity of the geodesic from $\mathbf{x}$ to $\mathbf{y}$.Therefore, $\beta'(1) = -log_{\mathbf{X}}(\mathbf{y})$ and it is sufficient to prove

$$\nabla f(\mathbf{x}) = 2\beta'(1). \tag{2.24}$$

Let $\tilde{\mathcal{V}}$ be the star-shaped neighborhood of the origin in $T_{\mathbf{y}}(\mathcal{M})$ and $(\mathcal{V}, \phi)$ be the corresponding normal neighborhood of $\mathbf{y} \in \mathcal{M}$. The function $f(\mathbf{x})$ measures the squared distance of $\mathbf{x}$ from $\mathbf{y}$. Since, the velocity of the geodesic is constant, the length of the geodesic is equal to the velocity along the geodesic

$$f(\mathbf{x}) = g_{\mathbf{y}}(log_{\mathbf{y}}(\mathbf{x}), log_{\mathbf{y}}(\mathbf{x})). \tag{2.25}$$

We now define, $\tilde{f} = f \circ \phi^{-1} : \mathbb{R}^m \rightarrow \mathbb{R}$. Given a point $\mathbf{u} \in \mathbb{R}^m$, $\phi^{-1}(\mathbf{u}) \in \mathcal{M}$ is a point on the manifold. By definition, we have $log_{\mathbf{y}}(\phi^{-1}(\mathbf{u})) = \sum_i u^i \mathbf{e}_i$, where u^i is the i-th coordinate of $\mathbf{u}$ and $\mathbf{e}_i$ is the orthonormal basis of $T_{\mathbf{y}}(\mathcal{M})$. Therefore,

$$\begin{aligned}
\hat{f}(\mathbf{u}) &= f(\phi^{-1}(\mathbf{u})) \\
&= g_{\mathbf{y}}(log_{\mathbf{y}}(\phi^{-1}(\mathbf{u})), log_{\mathbf{y}}(\phi^{-1}(\mathbf{u}))) \\
&= g_{\mathbf{y}}\left(\sum_i u^i \mathbf{e}_i, \sum_i u^i \mathbf{e}_i\right) \\
&= \sum_i (u^i)^2.
\end{aligned} \tag{2.26}$$

In the final step, we used the orthonormality of the basis $\mathbf{e}_i$. From [90, p.85], the j-th component of the gradient is given by

$$(\nabla f)^j = \sum_i g^{ij} \frac{\partial \hat{f}}{\partial u^i} = \frac{\partial \hat{f}}{\partial u^j} = 2u^j \tag{2.27}$$

where, we use the fact that the Riemannian metric matrix is the identity when expressed in normal coordinates. By definition, u^j is the component of the tangent $log_{\mathbf{y}}(\mathbf{x})$ along $\mathbf{e}_j$. Therefore, $\nabla f = 2\beta'(1)$ and (2.24) holds. The proof of the theorem follows.

2.3.4 Covariant Derivatives

Suppose the manifold under consideration is an open set $S \in \mathbb{R}^n$. For Euclidean space we have a global coordinate chart and the coordinate vector fields in this chart are the vector fields

$$\partial_i = \frac{d}{dx^i}, \quad i = 1, \ldots, n. \tag{2.28}$$

Let $V = \sum_i V^i \partial_i$ and $W = \sum_i W^i \partial_i$ be vector fields on this manifold. We can differentiate W with respect to V as

$$D_V W = \sum_i V(W^i)\partial_i \tag{2.29}$$

Therefore, the derivative of a vector field with respect to another vector field is itself a vector field. However, the above definition uses the Euclidean coordinates of the space and it is not clear how to extend this to arbitrary manifolds. This is done by defining a *connection* on the manifold which can intuitively be understood as introducing a correspondence between the tangent spaces at two points which are close to each other. Algebraically a connection D on a smooth manifold $\mathcal{M}$ is a function $D : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \rightarrow \mathfrak{X}(\mathcal{M})$ satisfying

$$D_{fU+hV} W = f D_U W + h D_V W \tag{2.30}$$

$$D_V(fW) = (Vf) D_V W + f D_V W \tag{2.31}$$

for $f, h \in \mathfrak{F}(\mathcal{M})$ are smooth functions and $U, V, W \in \mathfrak{X}(\mathcal{M})$ are smooth vector fields. The smooth vector field $D_V(W)$ is said to be the *covariant derivative* of W with respect to V. An important property of covariant derivatives is that the value of $D_V W$ at the point $\mathbf{x}$ depends only on the value of the vector field V at $\mathbf{x}$. For two vector fields U, V such that $U_{\mathbf{X}} = V_{\mathbf{X}}$, $D_V W|_{\mathbf{X}} = D_U W|_{\mathbf{X}}$. This is a consequence of (2.30) [90].

For a Riemannian manifold, there exists a unique connection D such that

$$[V, W] = D_V W - D_W V \tag{2.32}$$

$$Xg(V, W) = g(D_X V, W) + g(V, D_X W). \tag{2.33}$$

A connection that satisfies (2.32) is said to be *symmetric* and a connection that satisfies (2.33) is said to be *torsion-free*. Therefore, for a Riemannian manifold, there exists a unique symmetric, torsion-free connection and this is known as *Levi-Civita* connection of the Riemannian manifold [90].

2.3.5 Parallel Transport

Let $\alpha(t)$ be a curve on the manifold. For any vector field V we can compute the covariant derivative along the curve as $D_{\alpha'(t)}V$. The value of the covariant derivative $D_{\alpha'(t)}V$ only depends on the values of V along the curve [90].

A vector field V is said to be *parallel* to the curve α if $D_{\alpha'(t)}V = 0$ for $t \in [0, 1]$. For vector fields that are parallel to $\alpha(t)$ the value of the vector field at $\alpha(0)$ uniquely defines the value at every other point on the curve.

This property of being parallel establishes an explicit correspondence between the tangents at different points. Given two points $\mathbf{x}$ and $\mathbf{y}$ we can use any curve between $\mathbf{x}$ and $\mathbf{y}$ to parallel transport tangents from $\mathbf{x}$ to $\mathbf{y}$. The result is a tangent at $\mathbf{y}$, although the exact tangent obtained depends on the curve followed to go from $\mathbf{x}$ to $\mathbf{y}$. For Riemannian manifolds we use parallel transport along geodesics to build a unique correspondence between tangents at different points. This process of moving a tangent from $\mathbf{x}$ to $\mathbf{y}$ while being parallel to the geodesic is known as *parallel transport*.

2.3.6 Function Optimization

The idea of parallel transport is important since it is used to extend derivative based optimization methods such as Newton's iterations and conjugate gradient optimization to Riemannian manifolds [31, 106].

Gradient based minimization techniques proceed by iteratively improving the estimate of the position of the local minimum of a function f. Let $\mathbf{y}_j$ denote the estimate of the minimum after the j iterations. The $j + 1$-th iteration moves along the direction $-\nabla f(\mathbf{y}_j)$. This is known as the *search direction*. The estimate of the point is moved a certain distance along the search direction to get the next estimate of the minimum and these iterations are repeated till convergence. An important consideration in gradient minimization methods is how far along the negative gradient to move. Different algorithms propose different solutions to this. Newton iterations use the Hessian to build a quadratic local approximation of the function and use this to decide the step size. This method can be extended to manifolds by computing the Hessian over the manifold and building a local quadratic approximation like in Euclidean space.

Another popular gradient based optimization method is conjugate gradient minimization. In conjugate gradient, the search direction after j iterations is chosen to be orthogonal to all previous search directions. This requires the inner product of the current negative gradient with all previous search directions. Over manifolds, the $j+1$-th search direction is given by a tangent in $T_{\mathbf{y}_{j+1}}(\mathcal{M})$. The previous search directions lie in different tangent spaces at $T_{\mathbf{y}_j}(\mathcal{M}), T_{\mathbf{y}_{j-1}}(\mathcal{M})\ldots$ and it is not possible to compute an inner product directly. This problem is solved by using parallel transport to move previous search directions to $\mathbf{y}_{j+1}$ and then taking the inner product of the current negative gradient with the parallel transported versions of the previous search directions.

Both these optimization methods are known to have the same types of convergence behaviour as their Euclidean versions [31, 106]. We will later show an example of using conjugate gradient over a manifold for local function minimization in Chapter 4.

2.4 Types of Riemannian Manifolds

We briefly discuss the geometry of a few classes of Riemannian manifolds. Most frequently occurring manifolds in computer vision lie in one of these classes.

2.4.1 Lie Groups

A *Lie group* is a manifold which is also a group such that the group operation is an analytic mapping. The group operation gives Lie groups more algebraic structure than a manifold.

The most frequently occurring Lie groups are sets of matrices, *i.e.*, each element in the group is a matrix and the group operation is matrix multiplication. Such groups are called *matrix Lie groups* [97]. An alternative definition of matrix Lie groups is that they are closed subgroups of the general linear group $\mathbf{GL}(n)$, the group of $n \times n$

nonsingular matrices. Common examples of matrix Lie groups include

- *Special Orthogonal Group.* The special orthogonal groups, $\mathbf{SO}(n)$, is the set of rotations in $\mathbb{R}^n$. Elements of $\mathbf{SO}(n)$ are $n \times n$ orthogonal matrices.

- *Special Euclidean Group.* The special Euclidean group, $\mathbf{SE}(n)$, is the set of rigid transformations in $\mathbb{R}^n$. Matrices in $\mathbf{SE}(n)$ are $(n+1) \times (n+1)$ matrices of the form

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.34}$$

  where $\mathbf{R} \in \mathbb{R}^{n \times n}$ is orthogonal and $\mathbf{t} \in \mathbb{R}^n$.

- *Affine Group.* The affine groups $\mathbf{A}(n)$ consists of $(n+1) \times (n+1)$ matrices of the form

$$\begin{bmatrix} \mathbf{H} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.35}$$

  where $\mathbf{H} \in \mathbb{R}^{n \times n}$ is invertible and $\mathbf{t} \in \mathbb{R}^n$.

The tangent space at the identity of the group is known as the *Lie algebra* of the Lie group. Lie algebras are important since the tangent space at any point on the manifold can be expressed in terms of the Lie algebra [90, 97].

Lie groups are the most well known examples of Riemannian manifolds and the first nonlinear mean shift algorithm was proposed for Lie groups in [130]. Let *exp* and *log* be the matrix operators

$$exp(\boldsymbol{\Delta}) = \sum_{i=0}^{\infty} \frac{1}{i!} \boldsymbol{\Delta}^i \tag{2.36}$$

$$log(\mathbf{y}) = \sum_{i=1}^{\infty} \frac{(-1)^{i-1}}{i} (\mathbf{y} - \mathbf{e})^i \tag{2.37}$$

where, $\mathbf{e}$ is the identity matrix. These are standard matrix operators which can be applied to any square matrix and no subscript is necessary to define them. They should not be confused with the manifold operators, $exp_{\mathbf{x}}$ and $log_{\mathbf{x}}$ for Lie groups, which are given by

$$exp_{\mathbf{x}}(\Delta) = \mathbf{x}\ exp\left(\mathbf{x}^{-1}\Delta\right) \tag{2.38}$$

$$log_{\mathbf{x}}(\mathbf{y}) = \mathbf{x}\ log\left(\mathbf{x}^{-1}\mathbf{y}\right) \tag{2.39}$$

where, $\mathbf{y}$ is any point on the manifold and $\Delta \in T_{\mathbf{X}}(\mathcal{M})$ [90, Ch.11]. The distance function is given by

$$d(\mathbf{x}, \mathbf{y}) = \|log\left(\mathbf{x}^{-1}\mathbf{y}\right)\|_F \tag{2.40}$$

where, $\|.\|_F$ denotes the Frobenius norm of a matrix. This definition of d can be shown to be the distance corresponding to an inner product on $T_{\mathbf{X}}(\mathcal{M})$, as necessary for Riemannian manifolds [90, Ch.11].

There exist iterative techniques for computing the *exp* or *log* of general square matrices [49]. In practice, computational efficiency can be improved by taking advantage of the structure of the matrix Lie group under consideration. We show this with the example of the special orthogonal group, $\mathbf{SO}(3)$. Let $\mathfrak{so}(3)$ be the Lie algebra, *i.e.*, the tangent space at the identity. Elements of $\mathfrak{so}(3)$ are 3×3 skew-symmetric matrices of the form

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.41}$$

where the vector $\boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]$ is the axis of rotation and $\|\boldsymbol{\omega}\|$ is the magnitude of the rotation. The structure of $\mathbf{SO}(3)$ allows us to compute *exp* using the *Rodriguez* formula

$$exp([\boldsymbol{\omega}]_\times) = \mathbf{e_3} + \frac{sin\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|}[\boldsymbol{\omega}]_\times + \frac{1 - cos\|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|^2}[\boldsymbol{\omega}]_\times^2 . \tag{2.42}$$

where, $\mathbf{e_3}$ is the 3×3 identity matrix. The matrix logarithm can be computed by inverting the above equation [102].

2.4.2 Homogeneous Spaces

The *group action* of a group G on a manifold $\mathcal{M}$ is a smooth mapping from $G \times \mathcal{M}$ to $\mathcal{M}$. For $g \in G$ and $\mathbf{x} \in \mathcal{M}$, the mapping is written as $(g, \mathbf{x}) \to g \cdot \mathbf{x}$ and satisfies

$g \cdot (h \cdot \mathbf{x}) = (gh) \cdot \mathbf{x}$ and $e \cdot \mathbf{x} = \mathbf{x}$, where e is the identity of the group G. Just as matrices are transformations of vector spaces, group actions are transformations of manifolds.

The *orbit* of a point $\mathbf{x} \in \mathcal{M}$ is

$$O(\mathbf{x}) = \{g \cdot \mathbf{x} | g \in G\}. \tag{2.43}$$

The group action divides the manifold into a set of disjoint orbits. For example, consider the Euclidean plane $\mathbb{R}^2$ to be the manifold under the action of the rotation group $\mathbf{SO}(2)$. The orbits consist of circles centered at the origin.

If the whole manifold forms a single orbit, then $\mathcal{M}$ is said to be a *homogeneous space* and the action is said to be *transitive.* Therefore, the action is transitive, if for any two points $\mathbf{x}, \mathbf{y} \in \mathcal{M}$, there exists $g \in G$ such that $g \cdot \mathbf{x} = \mathbf{y}$. The Euclidean plane $\mathbb{R}^2$ under the action of the Euclidean motion group $\mathbf{SE}(2)$ would be a homogeneous space since for any two points on the plane, there exists some transformation mapping one to the other.

The *isotropy subgroup* of $\mathbf{x}$ is defined as

$$G_{\mathbf{X}} = \{g \in G | g \cdot \mathbf{x} = \mathbf{x}\}. \tag{2.44}$$

The isotropy subgroup is the set of elements which leave $\mathbf{x}$ fixed. In our previous example of $\mathbf{SE}(2)$ acting on $\mathbb{R}^2$, the isotropy subgroup of the origin would be the set of rotations. The isotropy subgroup of any other point on the Euclidean plane would be the set of rotations about that point.

If G is a Lie group and H is a subgroup of G, then the factor space G/H forms a manifold known as a *coset manifold.* The coset manifold is a homogeneous space of G with the natural group action

$$g \cdot kH = (gk)H \tag{2.45}$$

i.e., g acting on the coset of k gives the coset of gk.

The above result lets us represent a coset manifold as a homogeneous space. For a manifold with a transitive Lie group action, it is possible to reverse the process and think of it as a coset manifold. Let $\mathcal{M}$ be a homogeneous space under the Lie group action of G. For any arbitrary point $\mathbf{x} \in \mathcal{M}$, the manifold $\mathcal{M}$ can be identified with the homogenous space $G/G_{\mathbf{X}}$, where $G_{\mathbf{X}}$ is the isotropy subgroup of $\mathbf{x}$ [90, Ch.9].

The reason for using this characterization of coset manifolds is that they inherit a lot of their operators from the Lie group. Once the geometry of the Lie group is understood, the geometry of its coset manifolds can be expressed in terms of the geometry of the Lie group.

2.4.3 Grassmann Manifolds

A point on the *Grassmann manifold*, $\mathbf{G}_{n,k}$, represents a k-dimensional subspace of $\mathbb{R}^n$. In practice an element of $\mathbf{G}_{n,k}$ is represented by an orthonormal basis as a $n \times k$ matrix, i.e., $\mathbf{x}^T\mathbf{x} = \mathbf{e_k}$. Since many basis span the same subspace, this representation of points on $\mathbf{G}_{n,k}$ is *not* unique [31].

Consider any element $\mathbf{U}$ of the group $\mathbf{SO}(n)$. Its columns form an orthonormal basis of the space $\mathbb{R}^n$. A k-dimensional subspace can be obtained by taking the span of the first k-columns of $\mathbf{U}$ and this is an element of $\mathbf{G}_{n,k}$. However, rotations of the form

$$\begin{bmatrix} \mathbf{U}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_{n-k} \end{bmatrix} \tag{2.46}$$

where, $\mathbf{U}_k \in \mathbf{SO}(k)$ and $\mathbf{U}_{n-k} \in \mathbf{SO}(n-k)$, leave the subspace spanned by the first k-columns unchanged. Therefore, multiplication by elements of this form do not change the point in $\mathbf{G}_{n,k}$ and the set of all such rotations is equivalent to $\mathbf{SO}(k) \times \mathbf{SO}(n-k)$. We identify the Grassmann manifold $\mathbf{G}_{n,k}$ with the coset manifold $\mathbf{SO}(n)/(\mathbf{SO}(k) \times \mathbf{SO}(n-k))$.

The set of tangents at $\mathbf{x} \in \mathbf{G}_{n,k}$ consists of $n \times k$ matrices Δ satisfying $\mathbf{x}^T\Delta = 0$. For a tangent $\Delta \in T_{\mathbf{X}}(\mathcal{M})$ the exponential at $\mathbf{x}$ is

$$exp_{\mathbf{x}}(\boldsymbol{\Delta}) = \mathbf{x}\ \mathbf{v}\ cos(\mathbf{s})\ \mathbf{v}^T + \mathbf{u}\ sin(\mathbf{s})\ \mathbf{v}^T \tag{2.47}$$

where, $\mathbf{u}\ \mathbf{s}\ \mathbf{v}^T$ is the compact SVD of Δ and the *sin* and *cos* act element-by-element along the diagonal of $\mathbf{s}$ [31].

The *log* operator is the inverse of the *exp* operator. Let $\mathbf{x}$ and $\mathbf{y}$ be two points on the manifold $\mathbf{G}_{n,k}$. The logarithm of $\mathbf{y}$ at $\mathbf{x}$ is given by

$$log_{\mathbf{X}}(\mathbf{y}) = \mathbf{u}\; sin^{-1}(\mathbf{s})\; \mathbf{v}^T \tag{2.48}$$

where, $\mathbf{usd}^T = \mathbf{y} - \mathbf{xx}^T\mathbf{y}$ and $\mathbf{vcd}^T = \mathbf{x}^T\mathbf{y}$ is the generalized SVD with $\mathbf{c}^T\mathbf{c} + \mathbf{s}^T\mathbf{s} = \mathbf{e_k}$ and the sin^{-1} acts element-by-element along the diagonal of $\mathbf{s}$. It can be verified that this tangent does satisfy the two properties, $\mathbf{x}^T log_{\mathbf{X}}(\mathbf{y}) = 0$ and $exp_{\mathbf{X}}(log_{\mathbf{X}}(\mathbf{y})) = \mathbf{y}$ as required [31].

The distance between two points on the manifold is [1, 31]

$$d(\mathbf{x}, \mathbf{y}) = \|log_{\mathbf{x}}(\mathbf{y})\|_F \tag{2.49}$$

where $\|\cdot\|_F$ is the Frobenius norm.

2.4.4 Essential Manifold

The epipolar constraint encodes a relation between feature correspondences across two images of the same scene. In a calibrated setting, the epipolar constraint is parameterized by the *essential matrix*, a 3×3 matrix with certain algebraic properties [57, Sec.8.6.1]. The essential matrix represents the relative motion between two cameras [107], but due to the loss of depth information only the direction of translation can be recovered.

Let p and q be the normalized coordinates of corresponding points and $\mathbf{Q}$ be the essential matrix. The essential constraint is

$$p^T\mathbf{Q}q = 0. \tag{2.50}$$

Let $\mathcal{E}$ denote the *essential space*, the set of all essential matrices. The essential space is an algebraic variety [86, 107] and a manifold of dimension six. Essential matrices have some further algebraic properties. If $\mathbf{Q} = \mathbf{U\Sigma V}^T$ is the singular value decomposition of a 3×3 matrix $\mathbf{Q}$, then [57, Sec.8.6.1]

$$\mathbf{Q} \in \mathcal{E} \Leftrightarrow \mathbf{\Sigma} = diag\{\lambda, \lambda, 0\} \qquad \lambda \in \mathbb{R}^+ \tag{2.51}$$

i.e., an essential matrix has two equal, positive singular values and a zero singular value. Essential matrices are homogeneous quantities and scaling does not change the

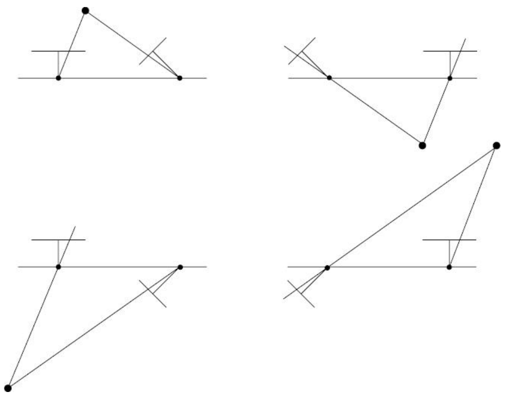


Figure 2.3: Four different camera geometries which give the same essential matrix. In each row the geometries differ by changing the sign of the direction of translation. Each column is a twisted pair. The image was taken from [57, p.241].

geometry. We use the scaling $\lambda = 1$ and define the *normalized essential space*, $\mathcal{E}_1$ as the set of 3×3 matrices with two unit singular values and one zero singular value

$$\mathbf{Q} \in \mathcal{E}_1 \Leftrightarrow \mathbf{\Sigma} = \mathbf{\Sigma}_1 \tag{2.52}$$

where, $\mathbf{\Sigma}_1 = diag\{1, 1, 0\}$.

The relative pose between two cameras can be recovered from their essential matrix except for two ambiguities. Firstly, since the epipolar geometry is a purely image based concept, there is no scale information and the baseline between the cameras can only be recovered up to a scale factor. Secondly, four different relative camera geometries give rise to the same essential matrix [57, p.241]. This is shown in Figure 2.3. Given an essential matrix the camera geometry can only be recovered up to this four-fold ambiguity. Usually, further image information is required to disambiguate the four geometries and choose the true geometry based on the *positive depth constraint.*

A common parametrization of the essential manifold is based on the fact that each relative camera geometry corresponds to a tangent of $\mathbf{SO}(3)$ with unit norm. The set of all tangents of a manifold itself forms a manifold known as the *tangent bundle.* Therefore, the essential manifold can be identified with the unit tangent bundle of $\mathbf{SO}(3)$ [80, 107]. Since each essential matrix corresponds to four different camera geometries, and each camera geometry corresponds to a different tangent of $\mathbf{SO}(3)$, this parametrization gives a four-fold covering of the essential manifold.

An alternate parametrization was proposed in [48]. This was based on a singular value decomposition of the essential matrix. This parametrization makes the essential manifold a homogeneous space under the action of the group $\mathbf{SO}(3) \times \mathbf{SO}(3)$.

Consider $\mathbf{Q} \in \mathcal{E}_1$ with the singular value decomposition $\mathbf{U}\mathbf{\Sigma}_1\mathbf{V}^T$, where $\mathbf{\Sigma}_1 = diag\{1, 1, 0\}$, $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and $det(\mathbf{U}), det(\mathbf{V}) = \pm 1$. As the third singular value is zero, we can change the sign of the third columns of $\mathbf{U}$ and $\mathbf{V}$ to ensure $det(\mathbf{U}), det(\mathbf{V}) = 1$ without changing the SVD.

Since $\mathbf{SO}(3)$ is a Lie group, the manifold $\mathbf{SO}(3) \times \mathbf{SO}(3)$ is also a Lie group with the topology and group operation inherited from $\mathbf{SO}(3)$ [97, Sec.4.3]. We define the mapping

$$\Phi : \mathbf{SO}(3) \times \mathbf{SO}(3) \rightarrow \mathcal{E}_1 \tag{2.53}$$

which maps $(\mathbf{U}, \mathbf{V}) \in \mathbf{SO}(3) \times \mathbf{SO}(3)$ to $\mathbf{U}\mathbf{\Sigma}_1\mathbf{V}^T \in \mathcal{E}_1$. The inverse mapping from $\mathcal{E}_1$ to $\mathbf{SO}(3) \times \mathbf{SO}(3)$ is not well defined as there is one degree of freedom in choosing the basis of the space spanned by the first two columns of $\mathbf{U}$ and $\mathbf{V}$. A rotation of the first two columns of $\mathbf{U}$ can be offset by a rotation of the first two columns of $\mathbf{V}$, such that $\mathbf{U}\mathbf{\Sigma}_1\mathbf{V}^T$ does not change. Consider the rotations $\mathbf{R}_1, \mathbf{R}_2$ such that

$$\mathbf{R}_1 = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & det(\mathbf{A}) \end{bmatrix} \qquad \mathbf{R}_2 = \begin{bmatrix} \pm\mathbf{A} & \mathbf{0} \\ \mathbf{0} & det(\mathbf{A}) \end{bmatrix}.$$

and $\mathbf{A}\mathbf{A}^T = \pm\mathbf{e_2}$. Then,

$$\mathbf{U}\mathbf{R}_1\mathbf{\Sigma}_1\mathbf{R}_2^T\mathbf{V}^T = \mathbf{U}\begin{bmatrix} \pm\mathbf{A}\mathbf{A}^T & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}\mathbf{V}^T = \pm\mathbf{U}\mathbf{\Sigma}_1\mathbf{V}^T \tag{2.54}$$

which leaves the essential matrix unchanged and Φ maps $(\mathbf{U}\mathbf{R}_1, \mathbf{V}\mathbf{R}_2) \in \mathbf{SO}(3)\times\mathbf{SO}(3)$ to the same point in $\mathcal{E}_1$.

Let $\mathbf{H}_\Phi$ be the group of transformations which leaves Φ invariant. It consists of elements which leave the third columns of $\mathbf{U}$ and $\mathbf{V}$ unchanged, and rotate the first two columns by angles which differ by $k\pi, k \in \mathbb{Z}$

$$\mathbf{H}_\Phi = \{(\mathbf{R}_1, \mathbf{R}_2) | \mathbf{R}_1, \mathbf{R}_2 \in \mathbf{S}_z, \mathbf{R}_1^T\mathbf{R}_2 = \mathbf{R}_z(k\pi)\} \tag{2.55}$$

where, $\mathbf{S}_z$ is the set of rotations around the z-axis and $\mathbf{R}_z(k\pi)$ denotes a rotation by $k\pi$ around the z-axis.

The manifold $\mathcal{E}_1$ is identified with the coset manifold $\mathbf{SO}(3) \times \mathbf{SO}(3)/\mathbf{H}_\Phi$, *i.e.*, elements of $\mathbf{SO}(3) \times \mathbf{SO}(3)$ which differ by group multiplication by an element in $\mathbf{H}_\Phi$ are considered to be the same on $\mathbf{SO}(3) \times \mathbf{SO}(3)/\mathbf{H}_\Phi$. Multiplication of $(\mathbf{U}, \mathbf{V})$ by elements of $\mathbf{H}_\Phi$ generates the equivalence class of $(\mathbf{U}, \mathbf{V})$, and all the elements in an equivalence class represent the same essential matrix.

The manifold $\mathbf{SO}(3) \times \mathbf{SO}(3)$ consists of two copies of $\mathbf{SO}(3)$ and the tangent space of $\mathbf{SO}(3) \times \mathbf{SO}(3)$ will consist of two copies of the tangent space of $\mathbf{SO}(3)$. Since $\mathbf{SO}(3)$ has three-dimensional tangent spaces, $\mathbf{SO}(3) \times \mathbf{SO}(3)$ will have six-dimensional tangent spaces. Consider $(\mathbf{U}, \mathbf{V}) \in \mathbf{SO}(3) \times \mathbf{SO}(3)$ and a tangent represented as a six vector

$$\Delta = \left[\mathbf{u}^T \ \mathbf{v}^T\right]^T \tag{2.56}$$

where, $\mathbf{u} = [u_x \ u_y \ u_z]^T$ and $\mathbf{v} = [v_x \ v_y \ v_z]^T$. The exponential for $\mathbf{SO}(3) \times \mathbf{SO}(3)$ is computed by performing the exponential of $\mathbf{SO}(3)$ twice, once each for $\mathbf{U}$ and $\mathbf{V}$

$$exp_{(\mathbf{U},\mathbf{V})}(\Delta) = (\ \mathbf{U}exp([\mathbf{u}]_\times), \mathbf{V}exp([\mathbf{v}]_\times)\ ) \tag{2.57}$$

where, the *exp* on the right represents the matrix exponential computed by the Rodriguez formula (2.42) and $[\cdot]_\times$ is defined by (2.41). The first three elements of the tangent vector correspond to $\mathbf{U}$ and the last three to $\mathbf{V}$. This ordering is equivalent to choosing a basis for the tangent space.

The tangent space of $\mathbf{SO}(3) \times \mathbf{SO}(3)$ can be divided into two complementary subspaces, the *vertical* and the *horizontal* space. The horizontal space contains tangents of the form

$$[u_x \quad u_y \quad u_z \quad v_x \quad v_y \quad -u_z], \qquad \|u_z\| < \pi/2. \tag{2.58}$$

The vertical space consists of tangents of the form

$$[0 \quad 0 \quad u_z \quad 0 \quad 0 \quad k\pi + u_z] \tag{2.59}$$

which lie in the Lie algebra of $\mathbf{H}_\Phi$ [31]. When $k = 0$, the vertical and horizontal spaces form complementary subspaces around the origin of the tangent space. Moving along geodesics defined by tangents in the vertical space is equivalent to multiplying by elements of $\mathbf{H}_\Phi$ and leaves the equivalence class unchanged. Vectors in the horizontal space are tangent to the equivalence class and all tangents of $\mathbf{SO}(3) \times \mathbf{SO}(3)/\mathbf{H}_\Phi$ must lie in the horizontal space of $\mathbf{SO}(3) \times \mathbf{SO}(3)$. Given a tangent in the horizontal space, its exponential can be computed like in (2.57) to get an element in another equivalence class (which will be a different essential matrix).

Let $(\mathbf{U}, \mathbf{V})$ and $(\hat{\mathbf{U}}, \hat{\mathbf{V}})$ represent two elements of $\mathbf{SO}(3) \times \mathbf{SO}(3)/\mathbf{H}_\Phi$. These can be any points in their respective equivalence classes. The logarithm operator for $\mathbf{SO}(3) \times \mathbf{SO}(3)/\mathbf{H}_\Phi$ should give a tangent in the horizontal space. To do this we first compute the logarithm on the manifold $\mathbf{SO}(3) \times \mathbf{SO}(3)$. Define

$$\delta\mathbf{U} = \mathbf{U}^T\hat{\mathbf{U}} \qquad \delta\mathbf{V} = \mathbf{V}^T\hat{\mathbf{V}}. \tag{2.60}$$

Taking the matrix logarithms of $\delta\mathbf{U}$ and $\delta\mathbf{V}$, and rearranging the elements into a six-vector, we get

$$[\delta u_x \; \delta u_y \; \delta u_z \; \delta v_x \; \delta v_y \; \delta v_z]^T \tag{2.61}$$

which lies in the tangent space of $\mathbf{SO}(3) \times \mathbf{SO}(3)$. Since $(\mathbf{U}, \mathbf{V})$ and $(\hat{\mathbf{U}}, \hat{\mathbf{V}})$ are arbitrary elements of their equivalence classes, it is not necessary that this vector lie in the horizontal space. We need to remove the component lying in the vertical space. Using Givens rotations [57, App.3] $\delta\mathbf{U}$ and $\delta\mathbf{V}$ are decomposed into rotations around the z-axis and rotations around axes in the xy-plane. Now, $(\mathbf{U}, \mathbf{V})$ is moved using z-rotations differing by $k\pi$, according to (2.57), so that on recomputing $\delta\mathbf{U}$ and $\delta\mathbf{V}$, they have opposite z-rotations less than $\pi/2$. This can be done in a single step and ensures that for the new $\delta\mathbf{U}$ and $\delta\mathbf{V}$, $\delta u_z \approx -\delta v_z$ up to a few degrees. Due to the nonlinearity of the manifold $\delta u_z = -\delta v_z$ will not hold exactly. This can be improved by moving $(\mathbf{U}, \mathbf{V})$ along tangents of the form

$$[0 \;\; 0 \;\; (\delta u_z + \delta v_z)/2 \;\; 0 \;\; 0 \;\; (\delta u_z + \delta v_z)/2]^T \tag{2.62}$$

and recomputing $\delta\mathbf{U}$ and $\delta\mathbf{V}$. The tangents of (2.62) lie in the vertical space and do not change the equivalence class of $(\mathbf{U}, \mathbf{V})$. After the initial step with Givens rotations,

$\delta u_z + \delta v_z$ is very small. Three or four iterations generally give an acceptable accuracy of the order of 10^{-4}. Once the *log* has been computed to obtain a tangent in the horizontal space, the intrinsic distance between $(\mathbf{U}, \mathbf{V})$ and $(\hat{\mathbf{U}}, \hat{\mathbf{V}})$ is given by the norm of the vector

$$d\left((\mathbf{U}, \mathbf{V}), (\hat{\mathbf{U}}, \hat{\mathbf{V}})\right) = \|\delta u_x \; \delta u_y \; \delta u_z \; \delta v_x \; \delta v_y\|_2. \tag{2.63}$$

2.4.5 Symmetric Positive Definite (SPD) Matrices

The set of $n \times n$ symmetric positive definite matrices forms a manifold known as the *symmetric manifold*, Sym_n^+. Recently, there has been a considerable amount of research aimed at understanding the geometry of this manifold due to the development of *diffusion tensor MRI* (DT-MRI) [6, 70], a widely used medical imaging method which measures the diffusivity of the water molecules in three dimensional space [72, 132]. The diffusivity is encoded as a 3×3 SPD matrix and the image is a 3D grid of 3×3 SPD matrices. The filtering of these images is an important step in their processing and requires an understanding of the noise model and the geometry of the manifold, Sym_3^+.

The computation of the matrix exponential and logarithm can be simplified in the case of the SPD matrices. Let $\mathbf{usu}^T$ be the singular value decomposition of a symmetric positive definite matrix $\mathbf{x} \in Sym_n^+$. Then the matrix logarithm is

$$log(\mathbf{x}) = \mathbf{u} log(\mathbf{s}) \mathbf{u}^T \tag{2.64}$$

where, the *log* acts on element-by-element along the diagonal of $\mathbf{s}$. Since the singular values of a SPD matrix are always positive, the log can act along the diagonal. The matrix exponential is computed similarly. Let Δ be a symmetric matrix with SVD given by $\Delta = \mathbf{udu}^T$. The matrix exponential becomes

$$exp(\boldsymbol{\Delta}) = \mathbf{u} exp(\mathbf{d}) \mathbf{u}^T. \tag{2.65}$$

where, the *exp* acts element-by-element along the diagonal of $\mathbf{d}$. It is easy to see that this form of the exponential is defined for any symmetric matrix and the result is a SPD matrix.

A metric for this manifold was first proposed in [40]. Later, in [93] it was shown that this metric is invariant to affine transformations and that Sym_n^+ is a Riemannian manifold. A different Riemannian metric for this manifold was proposed in [4], and this is the metric we use.

The idea behind the Riemannian metric of [4] is that the matrix *log* operator of (2.64) is a diffeomorphism over the space Sym_n^+. The range of the *log* is the space of $n \times n$ symmetric matrices (not necessarily positive definite). All operations are carried out by mapping SPD matrices to symmetric matrices using the *log* operator. Means and interpolation can be done in the space of symmetric matrices, which is a vector space, and mapped back to Sym_n^+ using the *exp* operator. It is shown in [4] that this corresponds to giving Sym_n^+ a Lie group structure. However, the group operation is *not* matrix multiplication and Sym_n^+ is not a matrix Lie group. The reason for using this Riemannian framework as opposed to the one proposed in [93] is that the numerical results are almost identical but the Lie group structure is more computationally efficient.

Let $\mathbf{x}$ and $\mathbf{y}$ be two $n \times n$ SPD matrices. The manifold logarithm is given by [4]

$$log_{\mathbf{X}}(\mathbf{y}) = log(\mathbf{y}) - log(\mathbf{x}). \tag{2.66}$$

Given a tangent $\mathbf{\Delta}$, the manifold exponential operator is

$$exp_{\mathbf{X}}(\mathbf{\Delta}) = exp(log(\mathbf{x}) + \Delta). \tag{2.67}$$

The distance between $\mathbf{x}$ and $\mathbf{y}$ is given by

$$d(\mathbf{x}, \mathbf{y}) = \|log(\mathbf{y}) - log(\mathbf{x})\|. \tag{2.68}$$

Chapter 3

Nonlinear Mean Shift

3.1 Introduction

The mean shift algorithm was first proposed in [43] and was further discussed in [16]. However, it became truly popular after [25, 26] used mean shift for color image segmentation and motion tracking. Since then, mean shift has been applied to many different problems including image segmentation [139, 140], tracking [8, 23, 32, 55] and robust fusion [15, 24].

Mean shift is essentially a clustering technique for finding meaningful centers of arbitrarily distributed points in a vector space. As it makes no assumptions about the nature of the distribution which generated the points, mean shift belongs to the class of nonparametric clustering methods. Mean shift has alternatively been shown to be equivalent to gradient ascent optimization of a kernel density [25], bounds optimization [35] and expectation-maximization [13]. In practice, the popularity of mean shift is due to the fact that the algorithm is easy to implement while exhibiting good convergence properties.

Here we present a generalized form of mean shift which can be used to cluster points which do not lie on a vector space. For example, consider points lying on the surface of a sphere. As we explain later, each iteration of mean shift requires the weighted sum of the data points around our current estimate of the mode. However, the weighted sum of points on the surface of the sphere does not lie on the sphere. In this case, it is possible to use our geometric intuition to estimate a mean, but geometric constraints often lead to much more complex curved surfaces where it is not possible to use our intuition.

An example of a problem which requires clustering over manifolds is the motion segmentation problem. It has been shown that robust estimation of motion parameters through voting based techniques, where votes from different motion parameter hypothesis are aggregated, give good results [87]. This would require the clustering of motion hypotheses over the space of possible motion parameters. To use mean shift for this purpose, we would need a formal definition of concepts such as the distance between motions and the average of a set of motions. In most cases the space of possible motion parameters is smooth and forms a Riemannian manifold.

Riemannian manifolds appear frequently in computer vision problems. In [80], geometric cost functions for reconstruction were treated as real valued functions on appropriate manifolds and minimized over these manifolds. The idea of treating motions as points on manifolds was used in [51] to smooth motions. In [92], it was shown that distances and distributions over manifolds can be defined based on the geometry they represent. The recent interest in the theory of manifolds is also due to the novel imaging data of new medical systems. Diffusion tensor images measure the diffusivity of water molecules and the smoothing of these images requires an understanding of the manifold nature of diffusion matrices [4, 72, 93, 132]. New image representations have also been developed for the statistical analysis of medical data using appropriate manifolds [38, 39, 28].

In [130] mean shift was extended to matrix Lie groups. A similar, but more general algorithm was proposed in [112], which could handle points lying on any Riemannian manifold, not necessarily Lie groups. Simultaneously, [7] proposed a slightly different mean shift algorithm for Grassmann manifolds.

We present a proper derivation of the nonlinear mean shift and discuss its theoretical properties. In deriving nonlinear mean shift we have concentrated on issues such as ease of implementation and computational efficiency. For example, a proper definition of kernel densities over manifolds would be too complicated. Consequently, we choose to define a reasonable approximation whose gradient can be easily computed.

3.2 The Original Mean Shift

The original mean shift algorithm is based on the theory of kernel density estimation. Here we describe the derivation of mean shift as in [25].

Let $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \ldots, n$ be n independent, identically distributed points generated by an unknown probability distribution f. The *kernel density estimate*

$$\hat{f}_k(\mathbf{y}) = \frac{c_{k,h}}{n} \sum_{i=1}^{n} k\left(\frac{\|\mathbf{y} - \mathbf{x}_i\|^2}{h^2}\right) \tag{3.1}$$

based on a *profile function* k satisfying $k(z) \geq 0$ for $z \geq 0$ is a nonparametric estimator of the density $f(\mathbf{y})$ at $\mathbf{y}$. The constant $c_{k,h}$ is chosen to ensure that $\hat{f}_k$ integrates to one.

Define $g(\cdot) = -k'(\cdot)$. Taking the gradient of (3.1) we get

$$\mathbf{m}_h(\mathbf{y}) = C\frac{\nabla \hat{f}_k(\mathbf{y})}{\hat{f}_g(\mathbf{y})} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\|\mathbf{y} - \mathbf{x}_i\|^2/h^2\right)}{\sum_{i=1}^{n} g\left(\|\mathbf{y} - \mathbf{x}_i\|^2/h^2\right)} - \mathbf{y} \tag{3.2}$$

where, C is a positive constant and $\mathbf{m}_h(\mathbf{y})$ is the *mean shift* vector. The expression (3.2) shows that the mean shift vector is proportional to a normalized density gradient estimate. The iteration

$$\mathbf{y}_{j+1} = \mathbf{m}_h(\mathbf{y}_j) + \mathbf{y}_j \tag{3.3}$$

is a gradient ascent technique converging to a stationary point of the density. Saddle points can be detected and removed, to obtain only the modes.

3.2.1 Mean Shift as Bounds Optimization

In the previous section mean shift was derived as a gradient ascent technique. The advantage of mean shift is that it avoids the computationally intensive line search step which other gradient ascent techniques require. The magnitude of the mean shift step adapts to the surrounding data and it can be shown that the mean shift iterations are guaranteed to converge to a local maxima of the kernel density [25].

An alternative view of mean shift was proposed in [35]. It was shown that for Epanechnikov kernels, the mean shift vector not only lies along the gradient but is in fact a Newton step. Furthermore, when using a general kernel, the mean shift step optimizes a lower bound on the kernel density function. This idea of mean shift as a variational bounds optimization was further developed in [105].

In [13], it was shown that for Gaussian kernels, the mean shift step is the same as Expectation-Maximization. In the M-step of the EM-algorithm a tight lower bound on the function is computed and in the E-step this bound is maximized. For non-Gaussian kernels, mean shift is equivalent to generalized EM.

All these approaches yield the same update rule when the data lies in Euclidean space, but for Riemannian manifolds this is not true. Generalizing each of these different algorithms to manifolds leads to different update rules. However, the reason for the widespread use of mean shift is due to its ease of implementation. It offers a simple iterative update rule with provable convergence behavior. We should take this into consideration when developing a mean shift update rule for manifolds. For example, using Newton iterations over manifolds to maximize a kernel density score would require the computation of Hessians. This step can be computationally expensive depending on the manifold. Therefore we propose a gradient based mean shift rule which does not require Hessian computation. However, the convergence properties of mean shift continue to hold for our nonlinear mean shift algorithm.

- Kernel density estimation over manifolds is more complex than in the Euclidean case. It requires the computation of a point dependent volume density function leading to a complex update rule.

- The weighted average of points on the manifold is well defined. Replacing a point by the weighted mean of the data points in a neighborhood still leads to convergence to the maxima of a cost function.

- For homogeneous spaces the nonlinear mean shift update rule is equivalent to expectation-maximization.

3.3 Nonlinear Mean Shift

The reason the previous mean shift algorithm is not directly applicable to manifolds is that manifolds are not vector spaces and the sum of points on the manifold, in general, does not lie on the manifold. Consequently, the mean shift vector of (3.2) is not valid. However, it is possible to define the weighted mean of points as the minimum of an appropriate cost function [68]. In this section, we use this concept to derive the mean shift vector as the weighted sum of *tangent vectors*. Since tangent spaces are vector spaces, a weighted average of tangents is possible and can be used to update the mode estimate. This method is valid over *any* Riemannian manifold.

3.3.1 Kernel Density Estimation over Riemannian Manifolds

Consider a Riemannian manifold with a metric d. Given n points on the manifold, $\mathbf{x}_i, i = 1, \ldots, n$, the kernel density estimate with profile k and bandwidth h is

$$\hat{f}_k(\mathbf{y}) = \frac{c_{k,h}}{n} \sum_{i=1}^{n} k\left(\frac{d^2(\mathbf{y}, \mathbf{x}_i)}{h^2}\right). \tag{3.4}$$

The bandwidth h can be included in the distance function as a parameter. However, we write it in this form since it gives us a handle to tune performance in applications. If the manifold is an Euclidean space with the Euclidean distance metric, (3.4) is the same as the Euclidean kernel density estimate of (3.1). The constant $c_{k,h}$ is chosen to ensure that $\hat{f}_k$ is a density, *i.e.*, the integral of $\hat{f}_k$ over the manifold is one.

Strictly speaking, $\hat{f}_k$ is not a kernel density. In Euclidean space the integral of the kernel is independent of the point at which it is centered. For a general Riemannian manifold, the integral of the kernel depends on the point at which it is centered. It is possible to ensure the integral of the kernel is the same irrespective of where it is centered by using the *volume density function* [91]. We do not do this, since the computation of the volume density function would limit the applicability of the algorithm to manifolds where explicit expressions for the volume densities are available. Also, the computation of the gradient would become very complicated. Therefore, we prefer to use the modified kernel density of (3.4). The kernel density we use here is similar to the one proposed in

[17, Ch.10] for Grassmann manifolds. We use the same expression for all Riemannian manifolds.

3.3.2 Mean Shift over Riemannian Manifolds

Calculating the gradient of $\hat{f}_k$ at $\mathbf{y}$, we get

$$\begin{aligned}\nabla \hat{f}_k(\mathbf{y}) &= \frac{1}{n}\sum_{i=1}^{n} \nabla k\left(\frac{d^2(\mathbf{y},\mathbf{x}_i)}{h^2}\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n} g\left(\frac{d^2(\mathbf{y},\mathbf{x}_i)}{h^2}\right)\frac{\nabla d^2(\mathbf{y},\mathbf{x}_i)}{h^2} \\ &= \frac{2}{n}\sum_{i=1}^{n} g\left(\frac{d^2(\mathbf{y},\mathbf{x}_i)}{h^2}\right)\frac{log_{\mathbf{y}}(\mathbf{x}_i)}{h^2} \qquad (3.5)\end{aligned}$$

where, $g(\cdot) = -k'(\cdot)$, and in the final step we use (2.22). The gradient of the distance is taken with respect to $\mathbf{x}$. Analogous to (3.2), define the nonlinear mean shift vector as

$$\mathbf{m}_h(\mathbf{x}) = \frac{\sum_{i=1}^{n} g\left(\frac{d^2(\mathbf{y},\mathbf{x}_i)}{h^2}\right) log_{\mathbf{y}}(\mathbf{x}_i)}{\sum_{i=1}^{n} g\left(\frac{d^2(\mathbf{y},\mathbf{x}_i)}{h^2}\right)}. \qquad (3.6)$$

All the operations in the above equation are well defined. The $log_{\mathbf{y}}(\mathbf{x}_i)$ terms lie in the tangent space $T_{\mathbf{y}}(\mathcal{M})$ and the kernel terms $g(d^2(\mathbf{y},\mathbf{x}_i)/h^2)$ are scalars. The mean shift vector is a weighted sum of tangent vectors, and is itself a tangent vector in $T_{\mathbf{y}}(\mathcal{M})$. The algorithm proceeds by moving the point along the geodesic defined by the mean shift vector. The noneuclidean mean shift iteration is

$$\mathbf{y}_{j+1} = exp_{\mathbf{y}_j}\left(\mathbf{m}_h(\mathbf{y}_j)\right). \qquad (3.7)$$

The iteration (3.7) updates $\mathbf{y}_j$ by moving along the geodesic defined by the mean shift vector to get the next estimate, $\mathbf{y}_{j+1}$. A mean shift iteration is started at each data point by initializing $\mathbf{y} = \mathbf{x}_i$. The inner loop then iteratively updates $\mathbf{y}$ till convergence. The complete algorithm is given below.

Algorithm: Mean Shift over Riemannian Manifolds

Given: Points on a manifold $\mathbf{x}_i, i = 1, \ldots, n$

```
for $i \leftarrow 1 \ldots n$
    $\mathbf{y} \leftarrow \mathbf{x}_i$
    repeat
        $\mathbf{m}_h(\mathbf{y}) \leftarrow \dfrac{\sum_{i=1}^{n} g\left(d^2(\mathbf{y}, \mathbf{x}_i)/h^2\right) log_{\mathbf{y}}(\mathbf{x}_i)}{\sum_{i=1}^{n} g\left(d^2(\mathbf{y}, \mathbf{x}_i)/h^2\right)}$
        $\mathbf{y} \leftarrow exp_{\mathbf{y}}\left(\mathbf{m}_h(\mathbf{y})\right)$
    until $\|\mathbf{m}_h(\mathbf{y})\| < \epsilon$
    Retain $\mathbf{y}$ as a local mode

Report distinct local modes.
```

Strong modes have high kernel density scores and a large number of iterations converge to these locations. Spurious modes, having low densities and few iterations converging to them, can be pruned to obtain only the strong modes. So, mean shift returns the number of modes and their locations.

3.4 Computational Details of Nonlinear Mean Shift

The nonlinear mean shift algorithm is valid for any Riemannian manifold. A practical implementation requires the computation of the exponential operator $exp_{\mathbf{x}}$ and the logarithm operator $log_{\mathbf{x}}$. The exponential and logarithm operators for commonly occurring manifolds were presented in the previous chapter. Table 3.1 lists these formulas for different manifolds.

For matrix Lie groups, the mean shift update proposed here is the same as the one

Table 3.1: Formulae for *exp* and *log* operators over different manifolds

	exp	*log*	$d(\mathbf{x}, \mathbf{y})$
Matrix Lie groups	(2.38)	(2.39)	(2.40)
Grassmann Manifolds	(2.47)	(2.48)	(2.49)
Essential Manifold	(2.57)	*	(2.63)
Symmetric Manifold	(2.67)	(2.66)	(2.68)

*Iterative procedure for logarithm over essential manifold is described in Section 2.4.4.

proposed in [112, 130]. However, in [112, 130] it was assumed that the relation between the Riemann distance and the *log* operator was an approximation. However, as we showed in Theorem 1, this relation is exact. Therefore the algorithm is provably convergent to a local maxima of the kernel density (3.4) which will be proved by Theorem 2.

For Grassmann manifolds the algorithm proposed here is different from [112]. Over there, an arc-length approximation to the Riemannian distance was used for computational purposes. This approximation was treated as a function of one of the points and the gradient was computed as in [31]. The approximation holds for points close to each other and in all experiments the algorithm converges quickly. However, theoretically the convergence of the approximation of [112] is not assured while the method proposed here is provably convergent.

In our implementation of mean shift over Sym_n^+, the *log* operator is used in a preprocessing step to map all matrices to their matrix logarithms. Mean shift is then carried out in the vector space of symmetric matrices and the modes are mapped back to Sym_n^+ by the *exp* operator. This is equivalent to performing nonlinear mean shift with the Lie group structure of Sym_n^+ [4]. However, the matrix logarithms and exponentials are used only once during the preprocessing and post-processing of data and the time taken for mean shift is much lower. The space of SPD matrices can be given another manifold structure and it is possible to do mean shift using this structure also. The details of this are given in [109].

3.5 Theoretical Properties

One of the reasons for the popularity of the original mean shift is the provable convergence of the mean shift iterations. In [25] it was shown that the iterations will converge to a local maxima of a kernel density. This is rather surprising since other gradient ascent methods such as Newton steps or conjugate gradient require a line search along the gradient to make sure that the iteration does not take a big step and miss the maximum. In the mean shift algorithms there is no line search. However, the mean shift step of (3.2) adapts itself to the data points surrounding it. In regions with a high density of points, the mean shift steps are small to make sure that it does not move too far and miss the maximum, while in regions with a low density of points the steps are much larger.

The nonlinear mean shift step of (3.6) ensures similar convergence properties. Let $\mathbf{y}_j, j = 1, \ldots$ be the successive estimates of the mode obtained through mean shift iterations given by (3.6) and (3.7). We have the following theorem.

Theorem 2. *If the kernel K has a convex and monotonically decreasing profile and the bandwidth h is less than the injectivity radius $i(\mathcal{M})$ of the manifold, the sequence $\{f(\mathbf{y}_j)\}_{j=1,2,\ldots}$ is convergent and monotonically non-decreasing.*

The proof proceeds is a manner similar to the proof for the Euclidean case [25]. Given the convexity of the profile we only need to show that each mean shift step minimizes a weighted error measure, which is the same as that minimized by weighted mean finding algorithms over manifolds. The necessary conditions for this have already been shown in [68].

Proof 2. To prove this theorem we use a result shown in [68]. Consider a set of points $\mathbf{x}_i, i = 1, \ldots, n$ lying on a Riemannian manifold. The *weighted Karcher mean* of these points with weights $w_i \in \mathbb{R}, i = 1, \ldots, n$ is defined as the points which minimize the cost function

$$C(\mathbf{y}) = \sum_{i=1}^{n} w_i d^2(\mathbf{y}, \mathbf{x}_i). \tag{3.8}$$

It was shown in [68] that if all the data points lie within an injectivity radius of each other, then the minimizer is unique and can be found using an iterative procedure. Let, $\mathbf{y}_j$ be the current estimate of the Karcher mean, then the updated estimate is given by

$$\mathbf{y}_{j+1} \quad = \quad exp_{\mathbf{y}_j}\left(\sum_{i=1}^{n} w_i log_{\mathbf{y}_j}(\mathbf{x}_i) \Bigg/ \sum_{i=1}^{n} w_i\right). \tag{3.9}$$

If $\mathbf{y}_{j+1}$ is obtained from $\mathbf{y}_j$ using the above rule, then $C(\mathbf{y}_{j+1}) \leq C(\mathbf{y}_j)$ and

$$\sum_{i=1}^{n} w_i(d^2(\mathbf{y}_j, \mathbf{x}_i) - d^2(\mathbf{y}_{j+1}, \mathbf{x}_i)) \geq 0. \tag{3.10}$$

Using this result we prove the convergence of the mean shift iterations. Each kernel is a real-valued function with a maximum value of one and the kernel density (3.4) is a sum of n such kernels. Since n is finite, the value of $f(\mathbf{y}_j)$ is bounded above. To prove convergence of the sequence $\{f(\mathbf{y}_j)\}_{j=1,2,...}$ it is sufficient to prove that

$$f(\mathbf{y}_{j+1}) \geq f(\mathbf{y}_j). \tag{3.11}$$

Using (3.4), we can write

$$f(\mathbf{y}_{j+1}) - f(\mathbf{y}_j) = \frac{c_{k,h}}{n} \sum_{i=1}^{n} \left[k\left(\frac{d_{j+1,i}^2}{h^2}\right) - k\left(\frac{d_{j,i}^2}{h^2}\right)\right] \tag{3.12}$$

where, we use the notation $d_{j+1,i} = d(\mathbf{y}_{j+1}, \mathbf{x}_i)$. Due to the convexity of the kernel we have

$$k(z_2) \geq k(z_1) + k'(z_1)(z_2 - z_1) \tag{3.13}$$

for any $z_1, z_2 \in \mathbb{R}$. Since, $g(\cdot) = -k(\cdot)$, we rewrite (3.13) as

$$k(z_2) - k(z_1) \geq g(z_1)(z_1 - z_2). \tag{3.14}$$

Using this identity in (3.12) for each of the n terms we get

$$f(\mathbf{y}_{j+1}) - f(\mathbf{y}_j) \geq \frac{c_{k,h}}{nh^2} \sum_{i=1}^{n} \left[g\left(\frac{d_{j,i}^2}{h^2}\right)(d_{j,i}^2 - d_{j+1,i}^2)\right]. \tag{3.15}$$

We now use (3.10) with the data points $\mathbf{x}_i, i = 1, \ldots, n$ and the weights $w_i = g(d_{j,i}^2/h^2)$. These weights change at each iteration, but for a single iteration they are constant and the inequality holds. In this case (3.9) is the same as the mean shift step (3.7) and therefore the right side of the above inequality if nonnegative. Therefore, (3.11) is true and the sequence of values $f(\mathbf{y}_j)$ is nondecreasing.

The only condition that needs to be verified is whether all the points lie within an injectivity radius of each other as required for the result of [68] to hold. This can be ensured by using a value of h less than the injectivity radius. In this case, the weights $g(d^2_{j,i}/h^2)$ are zero for all points further than h away and all points with nonzero weights will lie within an injectivity radius of each other as required.

3.5.1 Mean Shift and Expectation Maximization

In [13] it was shown that the original mean shift over vector spaces can be viewed as a special case of expectation maximization (EM). We adapt the proof from [13] to a general manifold. However, it is necessary to assume that there exists a transitive group action over the manifold. We give a brief outline of the proof showing that for a homogenous space, nonlinear mean shift with a Gaussian kernel is equivalent to EM.

Assume we are given a set of data points $\mathbf{x}_i, i = 1, \ldots, n$ lying on a manifold $\mathcal{M}$ endowed with a transitive group action. These data points define the kernel density

$$\hat{f}_k(\mathbf{y}) = \frac{c_{k,h}}{n} \sum_{i=1}^{n} k_g\left(\frac{d^2(\mathbf{y}, \mathbf{x}_i)}{h^2}\right). \tag{3.16}$$

where we use the Gaussian kernel $k_g(z) = e^{-z^2}$. Therefore, the kernel density is a mixture of Gaussian model defined over the manifold. The group action is used to define translated versions of this kernel density as

$$\hat{f}_k(\mathbf{y}|g) = \frac{c_{k,h}}{n} \sum_{i=1}^{n} k\left(\frac{d^2(g(\mathbf{y}), \mathbf{x}_i)}{h^2}\right). \tag{3.17}$$

where, g is an element of the group.

Given an observation $\hat{\mathbf{y}}$, we use the EM algorithm to find the most likely value of g. The hidden variables are denoted by $\mathbf{z}_i, i = 1, \ldots, n$ where $\mathbf{z}_i$ is the probability that $\hat{\mathbf{y}}$ was generated by the kernel centered at $\mathbf{x}_i$. Let g_j denote the estimate of g after the j-th iteration. The EM algorithm proceeds by building a tight lower bound for the log-likelihood of the conditional distribution in the E-step and then maximizing this lower bound in the M-step.

E Step: The log-likelihood is given by

$$\begin{aligned} log(p(\hat{\mathbf{y}}, g)) &= log\left(\sum_{i=1}^{n} p(\hat{\mathbf{y}}, \mathbf{z}_i, g)\right) \\ &\geq \sum_{i=1}^{n} p(\mathbf{z}_i|\hat{\mathbf{y}}, g_j) log\left[p(\mathbf{z}_i|g)p(\hat{\mathbf{y}}|\mathbf{z}_i, g)\right]. \end{aligned} \tag{3.18}$$

Assuming iid data, the expression of (3.18) can be further simplified to

$$\sum_{i=1}^{n} p(\mathbf{z}_i|\hat{\mathbf{y}}, g_j) log[p(\hat{\mathbf{y}}|\mathbf{z}_i, g)] + K \tag{3.19}$$

where the constant K is independent of g. The lower bound

$$Q(g|g_j) = \sum_{i=1}^{n} p(\mathbf{z}_i|\hat{\mathbf{y}}, g_j) log[p(\hat{\mathbf{y}}|\mathbf{z}_i, g)] \tag{3.20}$$

is maximized in the M-step.

M Step: The conditional probabilities $p(\hat{\mathbf{y}}|\mathbf{z}_i, g)$ are given by

$$p(\hat{\mathbf{y}}|\mathbf{z}_i, g) = k_g\left(\frac{d^2(g(\hat{\mathbf{y}}), \mathbf{x}_i)}{h^2}\right) \tag{3.21}$$

and the conditional probabilities $p(\mathbf{z}_i|\hat{\mathbf{y}}, g_j)$ are

$$p(\mathbf{z}_i|\hat{\mathbf{y}}, g_j) = k_g\left(\frac{d^2(g_j(\hat{\mathbf{y}}), \mathbf{x}_i)}{h^2}\right) \tag{3.22}$$

Therefore, the lower bound can be rewritten as

$$Q(g|g_j) = -\sum_{i=1}^{n} k_g\left(\frac{d^2(g_j(\mathbf{y}), \mathbf{x}_i)}{h^2}\right) d^2(g(\hat{\mathbf{y}}), \mathbf{x}_i). \tag{3.23}$$

Maximizing this lower bound is equivalent to minimizing the weighted sum on the right hand side, and this is the same weighted sum which is minimized at each step of the mean shift iteration. The only difference is that mean shift minimizes over the manifold while the EM procedure described above minimizes over the group. Since the group action is transitive the results of the two are equivalent.

It was also shown in [13] that Euclidean mean shift with a non-Gaussian kernel is equivalent to generalized EM. Similarly, it can be shown that for a for a Riemannian manifold with a transitive group action, nonlinear mean shift with a non-Gaussian kernel is equivalent to generalized EM.

3.6 Applications and Results

The nonlinear mean shift is used to develop a motion segmentation method which can recover both the number of motions and the motion parameters in a single step. This method can be used with a number of different motion models and examples of this are shown in Section 3.6.1. The motion segmentation algorithm can also be used for robust estimation as explained in Section 3.6.2. Another application of nonlinear mean shift is the discontinuity preserving filtering of non-vector valued images which is discussed in Section 3.6.3.

3.6.1 Motion Segmentation

The input to the algorithm consists of point matches belonging to different motions. Some of these correspondences may also be outliers. The algorithm proceeds in two stages. In the *first stage*, the matches are randomly sampled to generate minimal sets of points which define motion hypotheses. Such minimal sets are known as *elemental subsets*. The sampling and hypothesis generation can be improved by a *validation* step which reduces computation in the second stage [130]. In the validation step, a few correspondences are randomly selected and we check whether they satisfy the motion hypotheses. The hypotheses is retained only if at least one of these point satisfies the motion up to some predefined validation threshold.

In the *second stage*, the parameter estimates are clustered over the manifold of motion parameters using the algorithm proposed in Section 3.3.2. The number of dominant modes gives the number of motions in the data and the position of the modes corresponds to the motion parameters. Modes are considered not to be dominant if they do not have high kernel densities or do not have a large number of mean shift iterations converging to them.

The inliers for each mode are found based on the error residuals. This method is justified in Section 4.4.3. Briefly, for each motion parameter returned by the clustering, the residual errors of the inliers should be close to zero in the Euclidean space of the residuals. This will lead to a mode around the origin in the space of residuals. We use the original mean shift in the Euclidean space of the residual errors to find the basin of attraction of the mode at zero. Points with residuals in this basin of attraction are declared inliers. Since the inliers for each motion are decided independently, it is possible for a correspondence to be assigned to two motions. In such a case the tie is broken by assigning it to the motion which gives a lower error.

The performance of our algorithm is tested by verifying that the number of strong modes is equal to the number of motions present in the data. Since mean shift returns all local maxima of the kernel density estimate, for a data set with m motions the first m modes should clearly dominate the $(m+1)th$ mode, so that these extraneous modes can be pruned.

We quantitatively compare the result of the algorithm with a manual segmentation in two ways. Firstly, the classification of the correspondences is compared based on the number of misclassifications by the algorithm. Secondly, we compute the squared error using only the points declared inliers by the manual segmentation. Let $\mathbf{M}_j$ be the j-th mode returned by the clustering. If the correspondence $\mathbf{p}_i$ is an inlier for this motion, the residual $\mathbf{M}_j(\mathbf{p}_i)$ should be small. We measure the *residual squared error* of $\mathbf{M}_j$ as

$$\epsilon^j_{res} = \frac{1}{n_j} \sum_{i=1}^{n_j} |\mathbf{M}_j(\mathbf{p}_i)|^2 \tag{3.24}$$

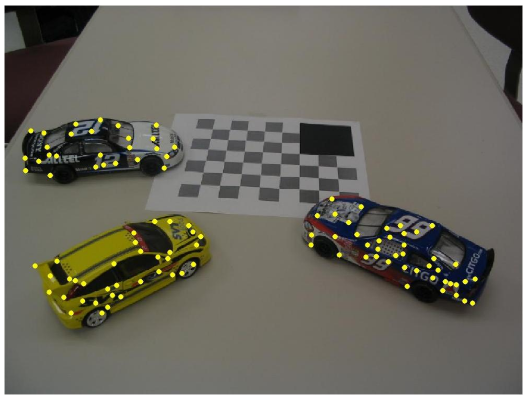
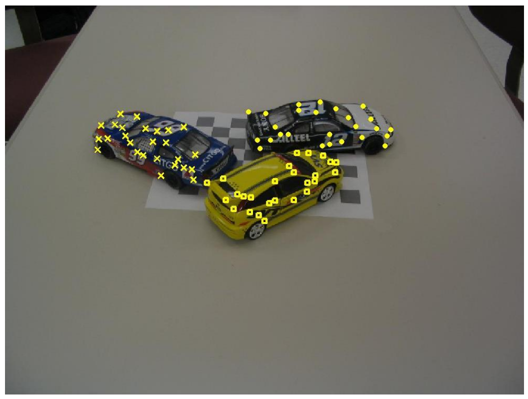
where the sum is taken only over correspondences which are inliers according to the manual segmentation and n_j is the number of inliers for the j-th motion. The lower limit for this error is

$$\epsilon^j_{LS} = \frac{1}{n_j} \sum_{i=1}^{n_j} \left|\hat{\mathbf{M}}_j(\mathbf{p}_i)\right|^2 \tag{3.25}$$

where, $\hat{\mathbf{M}}_j$ is the least squares estimate based on the inliers given by the manual segmentation. By definition, the least square estimate minimizes the squared error.

3D Translational Motion

Matched points across two views satisfy the epipolar constraint. If the camera is calibrated, the image coordinates can be converted to the normalized coordinates in 3D, and the epipolar constraint becomes the essential constraint. Furthermore, if the point

	mot.hyp.	kde
$\mathbf{M}_1$	107	0.0388
$\mathbf{M}_2$	173	0.0338
$\mathbf{M}_3$	102	0.0239
M_4	*66*	*0.0199*

	$\mathbf{M}_1$	$\mathbf{M}_2$	$\mathbf{M}_3$	*Out*	ϵ_{res}	ϵ_{LS}
$\mathbf{M}_1$	26	0	0	0	0.00112	0.00080
$\mathbf{M}_2$	0	26	0	0	0.00006	0.00006
$\mathbf{M}_3$	0	0	26	0	0.00179	0.00136
Out	1	0	0	23		

Figure 3.1: *3D Translational Motion. Mean shift over $G_{3,1}$.* In the left figure all the points are shown while on the right the inliers returned by the system. The table on the left contains the properties of the first four modes. Only the first three modes correspond to motions. The table on the right compares the results with the manual segmentations.

has undergone only translation with respect to the camera, the essential constraint is of the form [133]

$$\mathbf{t}^T (\mathbf{x}_1 \times \mathbf{x}_2) = 0 \tag{3.26}$$

where, $\mathbf{x}_1$ and $\mathbf{x}_2$ are the normalized coordinates in the two frames and $\mathbf{t}$ is the direction of translation in 3D. Since the constraint is homogeneous, the translation can only be estimated up to scale and $\mathbf{t}$ represents a line in $\mathbb{R}^3$ [133]. A line is a one-dimensional subspace of $\mathbb{R}^3$, so the translation is parameterized by the Grassmann manifold, $\mathbf{G}_{3,1}$. An elemental subset for a motion consists of two point matches and the hypotheses can be generated by a cross product.

The motion segmentation on a real data set with three translations is shown in Figure 3.1. A total of 102 corners were detected in the first frame and matched with corners in the second frame using the method of [47]. Points in the background were identified as having zero displacement and removed. On matching we obtain 27, 26 and 26 inliers for the motions and 23 outliers. These outliers are due to mismatches by the point matcher. We generated 500 motion hypotheses and clustered them on the manifold $\mathbf{G}_{3,1}$.

The results are tabulated in Figure 3.1. In the table on the left the number of hypotheses which converge to each mode and the kernel density at the mode are shown. Since the data set has three motions, there are three dominant modes with the fourth mode having fewer point converging to it. The segmentation results are on the right. Each row represents a motion and the row labeled *Out* represents outliers. The first four columns show the classification results. For example, the first row indicates that of the 26 points returned by the system as inliers for the first motion all 26 are inliers. The last row shows that of the 24 points declared to be outliers by the system one is actually an inlier for the first motion. Values along the diagonal are correctly classified, while off-diagonal values are misclassifications. The last two columns show the residual errors for our estimates ϵ, and for the least squares estimate, ϵ_{LS}. These residuals are the average reprojection errors in mm^2. Our algorithm's performance, with *no knowledge of the segmentation*, is comparable to manually segmented least squares estimates.

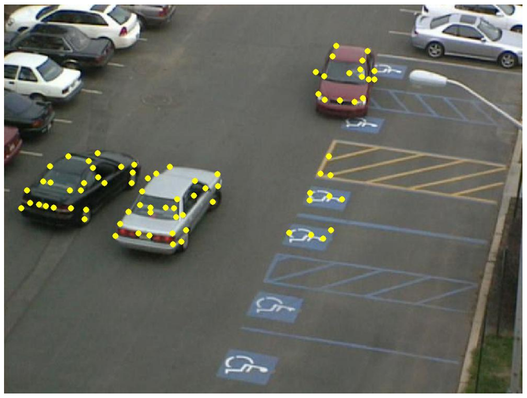
Affine Motion

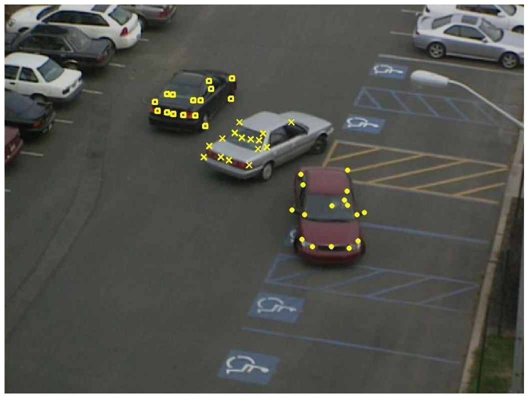
This example involves mean shift over the Lie group of affine image transformations. An affine image transformation is given by

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{3.27}$$

where, $\mathbf{A}$ is a nondegenerate 2×2 matrix and $\mathbf{b} \in \mathbb{R}^2$. The set of all affine transformations, $\mathbf{A}(2)$, forms a matrix Lie group. An affine transformation has 6 parameters and each point match gives 2 constraints. Each elemental subset, therefore consists of 3 point matches. The motion hypotheses are generated using least squares.

We used a data set of 80 corners matched across two frames with 3 independently moving bodies. The matching was done using [47]. Points on the background were identified as having zero displacement and removed. Some of the points on the background are occluded in the second image and are consequently mismatched. These points do not have zero displacements and survive as outliers in the data set. The motions contain 16, 18 and 16 inliers with 30 outliers. For clustering, 500 motion hypotheses were





	mot.hyp.	kde
$\mathbf{M}_1$	61	0.0550
$\mathbf{M}_2$	210	0.0547
$\mathbf{M}_3$	82	0.0468
M_4	11	0.0155

	$\mathbf{M}_1$	$\mathbf{M}_2$	$\mathbf{M}_3$	*Out*	ϵ_{res}	ϵ_{LS}
$\mathbf{M}_1$	16	0	0	0	2.7508	2.7143
$\mathbf{M}_2$	0	15	0	0	4.9426	4.6717
$\mathbf{M}_3$	0	0	15	0	3.2849	3.0860
Out	0	3	1	30		

Figure 3.2: *Affine motion segmentation. Mean shift over* $A(2)$. In the left figure all the points are shown, and on the right only the inliers are shown. The table on the left contains the properties of the first four modes. Only the first three modes correspond to motions. The table on the right compares the results with the manual segmentations.

generated. The results of the experiment are shown in Figure 3.2. The images and the tables display a similar analysis as in the previous figure. The residual errors are expressed in units of $pixel^2$.

Camera Pose Segmentation

The pose of a calibrated camera is the rigid body transformation from the world coordinate system to the camera coordinate system and the set of camera poses is the special Euclidean group, $\mathbf{SE}(3)$. We used the OpenCV camera calibration routines based on [142] to triangulate three different point clouds. A 10×7 checkerboard pattern was placed next to each of three objects and 25 images from different angles were taken for each object. A few of the images are shown in Figure 3.3. The OpenCV routine returns the internal camera parameters and the pose for each of the frames. We then used SIFT [78] to match features across the images and used the calibrated pose estimates to triangulate features in 3D. This gives us a data set of three different point clouds along with the SIFT descriptors for each feature. Each of the three point clouds is triangulated in a different 3D coordinate system.

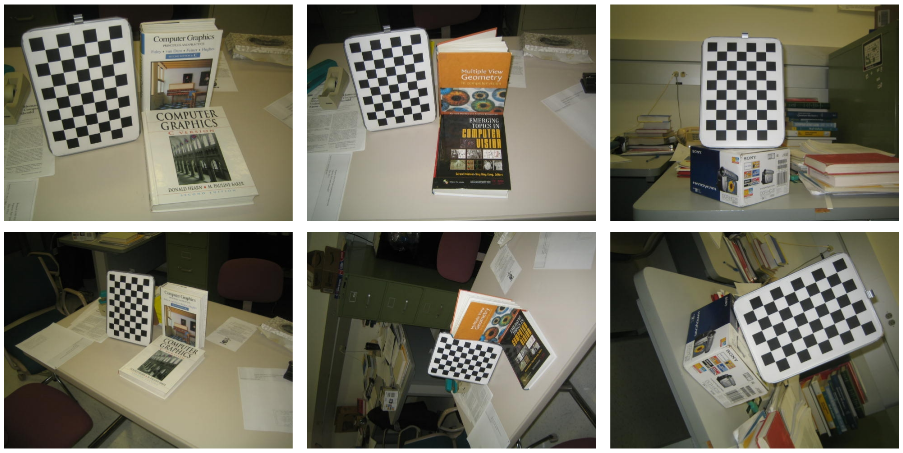
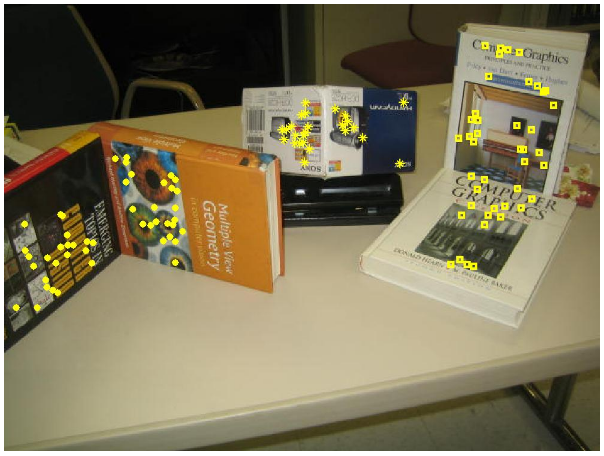


Figure 3.3: *Camera Pose Segmentation.* Images used to reconstruct 3D point clouds using the OpenCV implementation of the calibration technique of [142]

For segmentation, we use an image in which all three objects are present as shown in Figure 3.4. The relative pose of the camera with respect to the world coordinate systems of each of the objects is different. Therefore, the pose obtained using only the correspondences from each of the objects will be different. This property can be used to segment the three objects in the 2D image. Using SIFT we match the image features to the previously computed 3D features. This gives 3D to 2D correspondences lying across all three objects. Pose hypotheses are generated by the three-point method of [56]. Each elemental subset of three points gives up to *two* solutions. A 1000 hypotheses are clustered using mean shift over the Lie group $\mathbf{SE}(3)$ with a bandwidth of $h = 0.1$. The results of the segmentation are shown in Figure 3.4. The tables below the figure compare the results with the manual segmentation like in the previous sections with the residual errors are expressed in mm^2.

Multibody Factorization

Here we use mean shift over Grassmann manifolds. The positions of points tracked over F frames of an uncalibrated affine camera define a feature vector in $\mathbb{R}^{2F}$. For points sharing the same motion, these vectors lie in a four dimensional subspace of $\mathbb{R}^{2F}$, and for planar scenes this subspace is only three dimensional [117]. In a scene with multiple planar moving bodies, each motion defines a different subspace, which



	mot.hyp.	kde
$\mathbf{M}_1$	46	0.0388
$\mathbf{M}_2$	65	0.0387
$\mathbf{M}_3$	28	0.0223
M_4	16	0.0092

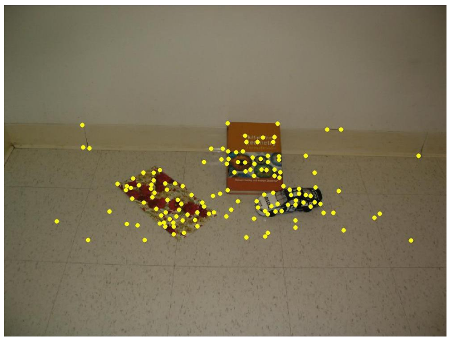
	$\mathbf{M}_1$	$\mathbf{M}_2$	$\mathbf{M}_3$	*Out*	ϵ_{res}	ϵ_{LS}
$\mathbf{M}_1$	42	0	0	1	2.46e-4	9.48e-5
$\mathbf{M}_2$	0	49	0	2	1.15e-4	3.90e-5
$\mathbf{M}_3$	0	0	25	0	4.63e-5	9.79e-6
Out	0	0	0	44		

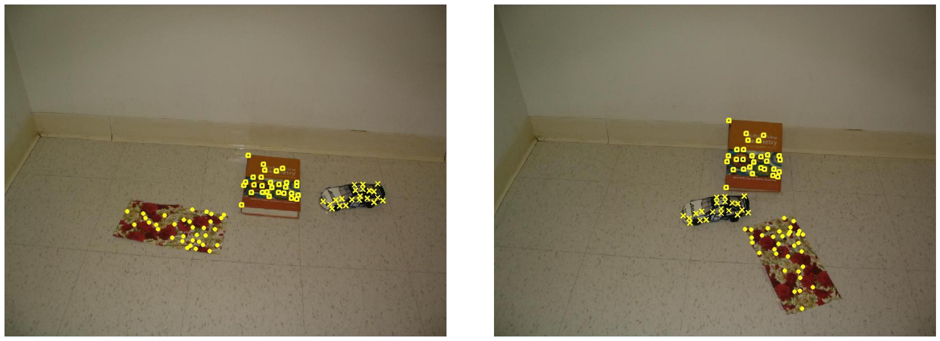
Figure 3.4: *Camera Pose Segmentation. Mean shift over* $\mathbf{SE}(3)$. The figure shows inliers for the different motions found. The table on the left contains the properties of the first four modes. Only the first three modes are valid motions. The table on the right compares the result with the manual segmentation.

can be represented by a point in the Grassmann manifold $\mathbf{G}_{2F,3}$. An elemental subset consists of the feature vectors defined by three points tracked across F frames. The basis is obtained through singular value decomposition.

The results of multibody factorization with three motions is shown in Figure 3.5. The system detected 140 corners in the first frame. Points on the background were identified as having zero displacement and removed. The rest of the corners were tracked across 5 frames, therefore, $F = 5$. The planar assumption holds due to negligible depth variation, and each motion defines a three-dimensional subspace of $\mathbb{R}^{10}$. The three motions contain 32, 21 and 29 points with 58 outliers. We generated 1000 hypotheses from these matches and clustered them on the manifold $\mathbf{G}_{10,3}$.

The results are organized like before with the residual errors are expressed in $pixel^2$. The kernel density at the fourth mode is an order of magnitude below the third mode. The classification results are nearly perfect. One outlier is misclassified as an inlier for the first motion.





	mot.hyp.	kde
$\mathbf{M}_1$	209	0.1315
$\mathbf{M}_2$	695	0.0830
$\mathbf{M}_3$	52	0.0165
M_4	12	0.0024

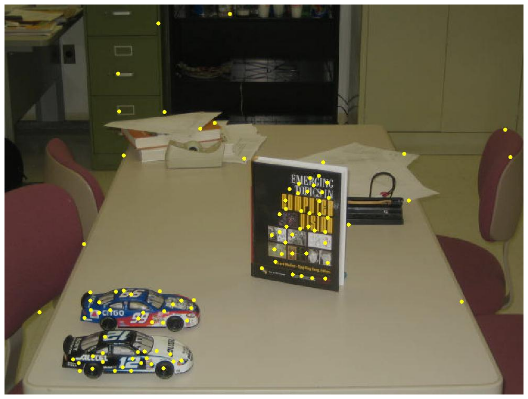
	$\mathbf{M}_1$	$\mathbf{M}_2$	$\mathbf{M}_3$	*Out*	ϵ_{res}	ϵ_{LS}
$\mathbf{M}_1$	32	0	0	1	7.0376	5.0193
$\mathbf{M}_2$	0	21	0	0	2.8520	0.7627
$\mathbf{M}_3$	0	0	29	0	4.2007	3.1648
Out	0	0	0	57		

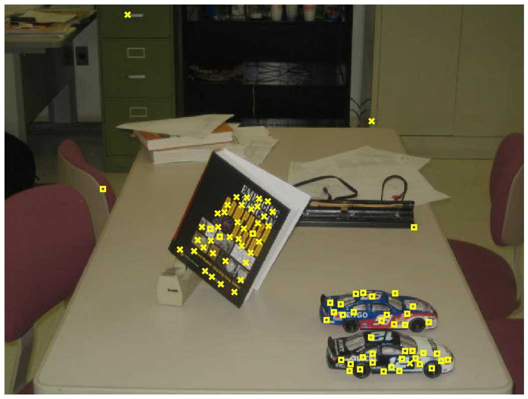
Figure 3.5: *Multibody Factorization. Mean shift over $G_{10,3}$.* The left figure shows the first frame with all the points which are tracked. The middle and right images show the second and fifth frames with only the inliers. The table on the left contains the properties of the first four modes. Only the first three modes correspond to motions. The table on the right compares the results with the manual segmentations.

Epipolar Segmentation

We do mean shift over the essential manifold. We calibrated a camera offline using the method of [142]. Two images of a scene with two moving bodies were taken. The points on each motion define an essential matrix due to their relative motion with respect to the camera. For hypotheses generation we used the method of [89]. Each elemental subset consists of five point and returns up to ten essential matrices. The hypotheses are clustered over the essential manifold using the theory developed in Section 2.4.4.

The two images used for motion segmentation are shown in Figure 3.6. The toy cars move together and the book has a separate motion. Using SIFT, and removing points





	mot.hyp.	kde
$\mathbf{M}_1$	459	0.0215
$\mathbf{M}_2$	409	0.0051
M_3	92	0.0026

	$\mathbf{M}_1$	$\mathbf{M}_2$	*Out*	ϵ_{res}	ϵ_{LS}
$\mathbf{M}_1$	36	1	2	5.31e-5	3.82e-5
$\mathbf{M}_2$	3	38	2	9.86e-4	1.64e-4
Out	0	3	15		

Figure 3.6: *Mean shift over the Essential Manifold.* The left figure shows the first frame with all the points which are matched. The right image shows the second frame with only the inliers returned by the segmentation. The table on the left contains the properties of the first three modes. Only the first two modes correspond to motions. The table on the right compares the results with the manual segmentations.

in the background as having zero displacement, we get 100 point matches with 39 on the book and 42 on the cars and 19 outliers according to the manual segmentation. We generated 1000 hypotheses and the mean shift was done with a bandwidth of $h = 0.001$. The clustering returns two dominant modes as expected. The first two modes are clearly more dominant than the third. Some of the outliers are misclassified as inliers since they satisfy the epipolar constraint. These results are tabulate in Figure 3.6 with the residual squared errors expressed in mm^2.

3.6.2 Robust Estimation

The same procedure that was used for motion segmentation can also be used for robust estimation. In this case, the motion hypotheses are expected to form a single mode in the parameter space. We integrated this robust estimator into the camera tracking system of [115]. A freely moving camera is viewing a scene with a specific world coordinate system. The aim is to robustly and efficiently compute the pose of the camera with respect to the world coordinate system. The camera is assumed to be calibrated offline and the required pose is the rigid body transformation from the world to camera coordinate systems.

The system is initialized using a set of easily identifiable markers placed in the scene which define a world coordinate system. Initially, the pose is estimated from these markers and there are no outliers. Features in the rest of the scene are triangulated using these pose estimates. The camera is then allowed to move freely without being required to keep the markers in view. Triangulated features are used to estimate pose while further features are constantly reconstructed. At this stage, the robust estimator is required to prevent mismatches in the image tracking from leading to erroneous pose estimates. In practice, a robust pose estimator is not sufficient for good results. Each robust fit is used to remove outliers and the final pose is estimated using only the inliers. Given a set of 3D world to 2D image point correspondences, we use the three-point algorithm of [56] to generate pose hypotheses. The hypotheses are clustered using mean shift over $\mathbf{SE}(3)$.

The mean shift estimator also allows us to take advantage of the continuity of the camera movement. Since the pose estimates of two consecutive frames will not be very different from each other, rather than starting a mean shift iteration at each hypothesis, we only try to find the mode closest to the previous frame's pose. Therefore a single mean shift iteration is initialized at the previous pose estimate. The point of convergence is taken as the next robust pose estimate. This also reduces the computation since we only need to do a single set of mean shift iterations rather than start an iteration at each data point.

Mean Shift versus RANSAC

We outline a simple proof of why mean shift performs better than hypothesis-and-test algorithms. Assume the data consists only of noisy inliers. With perfect data all hypotheses will lie exactly at the true pose. For noisy data, the hypotheses $\mathbf{P}_i, i = 1, \ldots, n$ are distributed around the true pose.

We assume the algorithm for hypothesis generation is unbiased. The generated hypotheses will form a unimodal distribution with the mode at the true pose. This

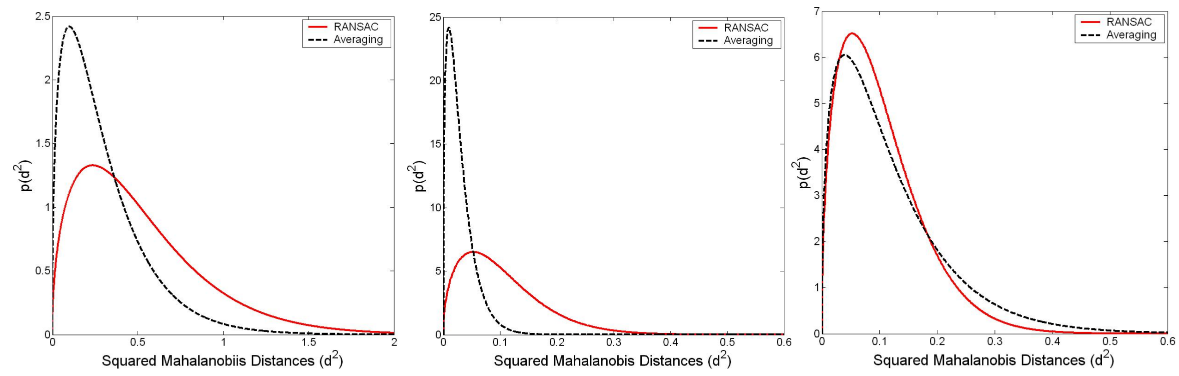


Figure 3.7: Comparison of the error densities for RANSAC and averaging as given by (3.28) and (3.29). (a) $n = 10$ for both curves. (b) $n = 100$ for both curves. (c) $n = 100$ for RANSAC and $n = 25$ for averaging.

mode is modeled as a Gaussian with mean at the true pose $\mathbf{P}_o$ and covariance $\mathbf{\Sigma}$. Since $\mathbf{SE}(3)$ is a 6-dimensional manifold in 12-dimensional space, Σ is a 12×12 matrix of rank 6 [65]. The squared Mahalanobis distances of the hypotheses from $\mathbf{P}_o$ forms a χ^2 distribution with 6 degrees of freedom (dof). Let f and F be the density and distribution functions of a 6 dof χ^2 distribution. Let $\mathbf{P}_r$ be the RANSAC result and $\mathbf{P}_a$ be the average of n hypotheses. We compare the two estimates based on their Mahalanobis distances from $\mathbf{P}_o$.

RANSAC will always return one of the generated hypotheses. Ideally, it will return the hypothesis with the lowest Mahalanobis distance to $\mathbf{P}_o$. The probability of the lowest Mahalanobis distance being d and all the others being greater than d is

$$p(\|\mathbf{P}_r - \mathbf{P}_o\|^2_{\mathbf{\Sigma}} = d^2) = nf(d^2)(1 - F(d^2))^{n-1} \; . \tag{3.28}$$

The mean of n Gaussian variables is a Gaussian random variable with the same mean but an n times less covariance. Therefore, $\mathbf{P}_a$ is a Gaussian random variable with mean $\mathbf{P}_o$ and covariance Σ/n. Consequently, $n\|\mathbf{P}_a - \mathbf{P}_o\|^2_{\mathbf{\Sigma}}$ is a 6 dof χ^2 variable and this gives

$$p(\|\mathbf{P}_a - \mathbf{P}_o\|^2_{\mathbf{\Sigma}} = d^2) = nf(nd^2) \; . \tag{3.29}$$

The distributions for $n = 10$ and $n = 100$ are compared in the first two images of Figure 3.7. The averaged estimates are closer to the true pose, and as n increases this difference becomes more obvious. Therefore, averaging requires fewer hypotheses to perform as well as RANSAC.

In the presence of outliers, the hypotheses will no longer form a unimodal distribution around the true pose. However, the pose estimates generated using only inliers will still be distributed in the same way. Ideally, RANSAC will return the closest of these estimates, and the above analysis for RANSAC still holds with n begin the number of hypotheses generated using only inliers. To prevent outlier hypotheses from affecting the averaging, the averaging needs to be robust. Mean shift (with the Epanechnikov kernel) is the mean of all the points lying within the basin of attraction [25]. For an appropriately chosen bandwidth, the mean shift estimate will be the average of all the inlier hypotheses and the distance of this value from the true pose will follow the distribution (3.29). Since averaging requires fewer hypotheses for the same level of performance and the major bottleneck in the hypothesis-and-test procedure is the generation of the hypotheses, less time is spent removing outliers.

In practice, the above assumptions may not hold. The hypotheses need not be normally distributed, although for low noise this does not lead to serious problems. More importantly, the bandwidth of the mean shift is usually conservative and not all inlier hypothesis are averaged. Therefore, the parameter n differs for mean shift and RANSAC. In the third curve of Figure 3.7, we compare the RANSAC error density of (3.28) for $n = 100$ and the averaging error density of (3.29) for $n = 25$. As these densities are comparable, mean shift needs to average only 25 good hypotheses to be as good as RANSAC with 100 inlier hypotheses.

Camera Tracking System

The workspace scene from [115] was used to test our system. An image of this scene and the camera path and the reconstructed point cloud for a sequence are shown in Figure 3.8. Initially the camera is moved in front of the markers to allow scene features to be reconstructed. This is the set of frames lying along a line in the top left of the image. Later, the camera is allowed to move away from the markers and the robust estimator is used.

We ran our experiments on a $2.4GHz$ Pentium 4 machine. RANSAC requires 100

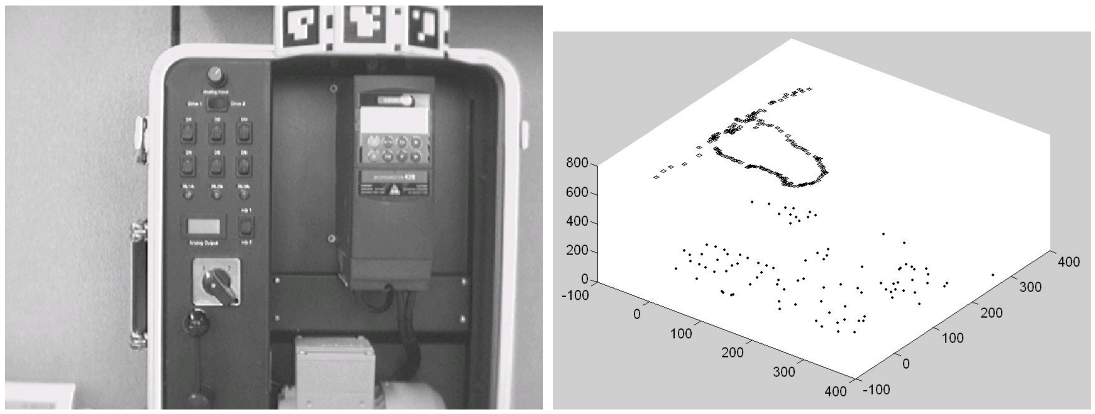


Figure 3.8: Results of the camera tracking. The scene used is shown on the left. The reconstructed point cloud and camera frames are on the right.

hypothesis and takes 0.4ms to process them. Each hypothesis generation takes 0.05ms leading to a total of 5.4ms for the robust estimator. The mean shift estimator requires 50 hypothesis for similar performance and takes 1.2ms, on the average, to find the mode. This gives a total time of 3.7ms for the mean shift estimator.

3.6.3 Discontinuity Preserving Filtering

The original mean shift has been used for the discontinuity preserving filtering of color images [19, 25]. This algorithm was extended to manifold valued images in [114].

The *image* $\mathbf{I}$ is considered to be a mapping on a n-dimensional lattice which assigns a value to each lattice point. Typically, $n = 2$ or 3. At each location $\mathbf{z}_i$, the data values $\mathbf{I}(\mathbf{z}_i)$ are assumed to lie on a Riemannian manifold, $\mathcal{M}$. A pixel $\mathbf{I}(\mathbf{z}_i)$ along with its location $\mathbf{z}_i$ is considered as a single data point $\mathbf{x} = (\mathbf{z}, \mathbf{I}(\mathbf{z}))$, in the *joint domain* $\mathbb{R}^n \times \mathcal{M}$.

We do mean shift in this joint space to cluster the pixels. Consider an iteration starting at the point $\mathbf{x}_i = (\mathbf{z}_i, \mathbf{c}_i)$, $\mathbf{c}_i = \mathbf{I}(\mathbf{z}_i)$. Let this iteration converge to $(\hat{\mathbf{z}}_i, \hat{\mathbf{c}}_i)$. In the filtered image $\mathbf{I}_f$, we set $\mathbf{I}_f(\mathbf{z}_i) = \hat{\mathbf{c}}$. The profile in the joint domain is the product of a *spatial profile* defined on the Euclidean part of the joint domain and a *parameter profile* defined on the manifold, as

$$k(\mathbf{x}, \mathbf{x}_i) = k_s\left(\frac{\|\mathbf{z} - \mathbf{z}_i\|^2}{h_s^2}\right) k_p\left(\frac{d^2(\mathbf{c}, \mathbf{c}_i)}{h_p^2}\right) \tag{3.30}$$

where $\mathbf{x} = (\mathbf{z}, \mathbf{c})$ and $\mathbf{x}_i = (\mathbf{z}_i, \mathbf{c}_i)$ are point in the joint domain. The bandwidth in

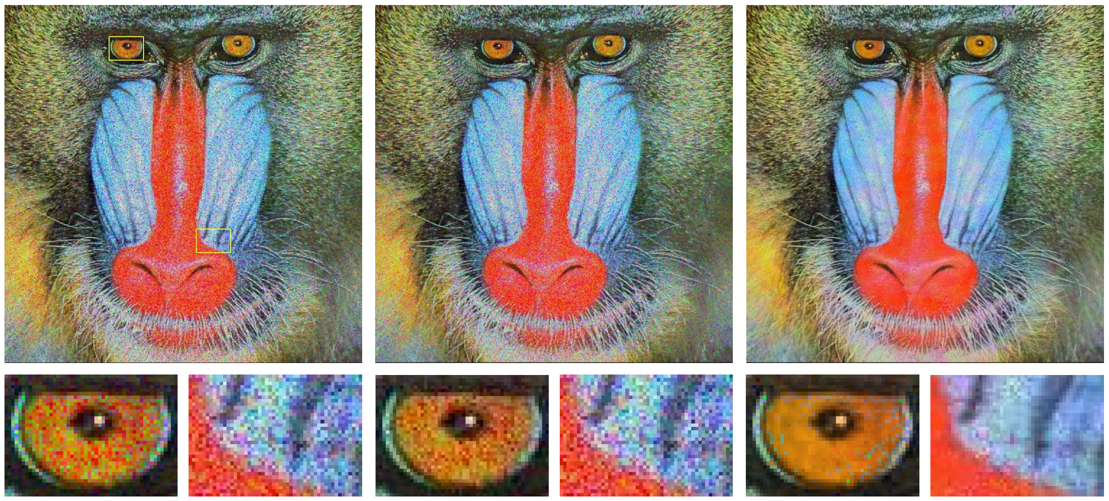


Figure 3.9: *Chromatic Noise Filtering.* The *baboon* image corrupted with chromatic noise is shown on the left. The results of using standard mean shift filtering with EDISON are in the middle and the results of our method are on the right.

the joint domain consists of a spatial bandwidth h_s and a parameter bandwidth h_p. In practice, we use a truncated normal kernel and the performance of the algorithm can be controlled by varying h_p and h_s.

To optimize performance, we used the heuristic suggested in [19] and used in the EDISON system. The filtering step was not applied to pixels which are on the mean shift trajectory of another (already processed) pixel. These pixels were directly associated with the mode to which the path converged. The approximation does not noticeably change the filtered image but reduces processing time.

Chromatic Noise Filtering

Chromatic image noise affects the direction (chromaticity) of the color vector and not its intensity. The direction of a 3D vector can be represented by a unit vector in 3D and these form the *Grassmann manifold*, $\mathbf{G}_{,3}1$. By filtering chromaticity we obtain better results than original mean shift which smooths chromaticity and brightness.

The results for the *baboon* image are shown in Figure 3.9. Chromatic noise of standard deviation 0.2 was added to the original image. The original mean shift image filtering algorithm from EDISON, with spatial bandwidth $h_s = 11.0$ and color bandwidth $h_p = 10.5$, was used to get the middle image. Using a larger h_p leads to oversmoothing

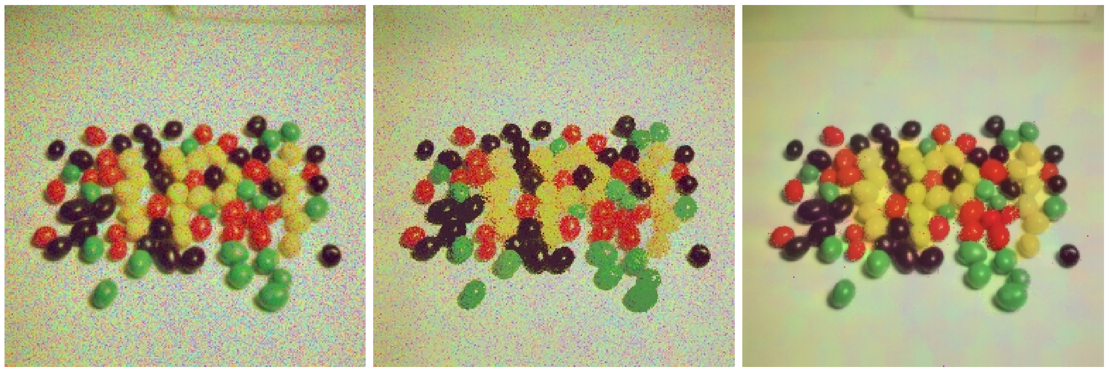


Figure 3.10: *Chromatic Noise Filtering.* The *jellybeans* image corrupted with chromatic noise is shown on the left. The results of using standard mean shift filtering with EDISON are in the middle and the results of our method are on the right.

and using smaller values does not change the image much. Our algorithm was run with $h_s = 11.0$ and $h_p = 0.3$ to get the image on the right. To clearly illustrate the difference in the results, two image regions, outlined in yellow in the input image, are shown in close-up. Our filtering is clearly better than EDISON.

The results for the *jellybeans* image are shown in Figure 3.10. The image is corrupted with chromatic noise of standard deviation 0.2. The original mean shift image filtering algorithm from EDISON, with spatial bandwidth $h_s = 11.0$ and color bandwidth $h_p = 10.5$, was used to get the middle image. Our algorithm was run with $h_s = 11.0$ and $h_p = 0.3$ to get the image on the right. Again, our filtering is better than EDISON due to the averaging of the right noise model.

DT-MRI Filtering

DT-MRI filtering involves mean shift over $\mathbb{R}^3 \times Sym_3^+$. Our real data set is a DTI of the human heart obtained from [59]. The lattice size is $128 \times 128 \times 67$ and we ran the smoothing with bandwidth values $h_s = 9.0$ and $h_p = 1.0$. For visualization purposes, each 3×3 diffusion matrix is converted to some scalar value and planes of the 3D lattice are drawn. Here, we use the *fractional anisotropy* [129]

$$\sqrt{\frac{3}{2}\frac{(\lambda_1 - \bar{\lambda})^2 + (\lambda_2 - \bar{\lambda})^2 + (\lambda_3 - \bar{\lambda})^2}{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \tag{3.31}$$

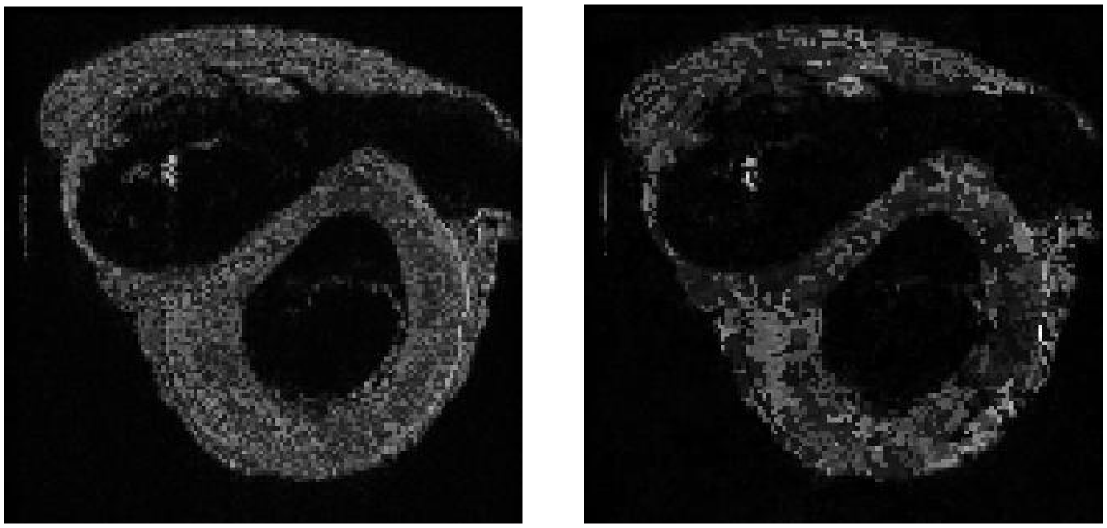


Figure 3.11: Real DTI data of a human heart before and after smoothing. The jitter in the top image is due to noisy voxels having different anisotropies from their surroundings. These are removed by the smoothing and more continuous regions of uniform anisotropy are visible below.

where, λ_1, λ_2 and λ_3 are the eigenvalues and $\bar{\lambda} = (\lambda_1 + \lambda_2 + \lambda_3)/3$. The fractional anisotropy for a particular plane $z = 47$ is shown in Figure 3.11.

Chapter 4

Projection Based M-Estimators

Regression algorithms estimate a parameter vector given a data set and a functional relation between the data and the parameters. If some of the data points do not satisfy the given relation they are known as *outliers*, while points which do satisfy it are known as *inliers*. Outliers interfere with the regression and lead to incorrect results, unless they are appropriately accounted for. *Robust regression* algorithms perform regression on data sets containing outliers without a significant loss of accuracy in the parameter estimates.

In vision applications, outliers almost always occur and any system which aims to solve even simple visual tasks must address this problem. The most widely used robust algorithm in computer vision is *Random Sample Consensus* (RANSAC) [37]. The popularity of the original RANSAC algorithm is due to its ease of implementation. However, RANSAC suffers from a number of drawbacks which make it inappropriate for some real applications.

Here we discuss an important obstruction to applying RANSAC in practice which has largely been ignored till recently, namely the sensitivity to *scale*. Scale is the level of additive noise which corrupts the *inliers*. RANSAC requires an estimate of this value to be specified by the user and the performance of RANSAC is sensitive to the accuracy of the scale estimate. Using a low value of the scale leads to rejecting valid inlier data, while using large estimates of the scale lets the outliers affect the parameter estimate.

We develop a robust regression algorithm known as the projection based M-estimator (pbM), which *does not require a user specified scale estimate*. This is achieved by establishing a connection between regression and the theory of *kernel density estimation*. The pbM algorithm was initially proposed in [14] for robust regression with homoscedastic data vectors, *i.e.*, data vectors which are corrupted by additive noise with the same covariance. Since then pbM has undergone changes in the cost function [110] and variations of pbM have been developed to handle more complex problems such as heteroscedastic data [111] and multiple linear constraints [113].

The pbM uses a modification of a robust M-estimator cost function to develop data driven scale values. Usually, in regression problems, all the data points are assumed to be corrupted by additive noise with the same covariance. Such data sets are *homoscedastic*. The pbM algorithm is extended to two scenarios both of which are more complex than the usual homoscedastic robust regression. Firstly, we consider *heteroscedastic* data, where each inlier data point is allowed to be corrupted by an additive noise model with a different covariance. As has been shown before [84], even in the presence of low-dimensional homoscedastic data, the nonlinear functional relations which occur in 3D-vision lead to heteroscedastic data vectors for regression. Secondly, we address the problem of *multiple regression*, where each data point is required to satisfy *multiple* linearly independent constraints. This problem is equivalent to robust subspace estimation or robust principal component analysis [3, 12] and is of practical importance to solve problems such as factorization [94, 118].

4.1 Previous Work

Random sample consensus (RANSAC) is the most widely used robust estimator in computer vision today. RANSAC was proposed in [37] and has since then been applied to a wide range of problems including fundamental matrix estimation [122], trifocal tensor estimation [123], camera pose estimation [88] and structure from motion [88]. Other applications of RANSAC can be found in [57]. RANSAC has been used extensively and has proven to be better than various other robust estimators. Some of these estimators, such as LMedS [98], were developed in the statistics community, but were found to not give good results for vision applications [108].

The RANSAC algorithm proceeds by repeatedly hypothesizing possible parameter estimates and then checking for the number of inliers based on the residual error. The hypothesis with the maximum number of inliers is declared the true estimate. The decision of whether a point is an inlier or not is based on a user defined scale estimate.

4.1.1 RANSAC and Robust Regression

Improvements have been proposed to the basic RANSAC algorithm of [37]. These improvements can broadly be divided into two classes

- changes to the cost function
- changes to the sampling.

In [123], it was pointed out that in the RANSAC cost function, all inliers score zero uniformly and all outliers score a constant penalty. Better performance was obtained with a cost function where the inlier penalty is based on its deviation from the required functional relation, while outliers scored a constant penalty. The method is known as MSAC (M-estimator sample consensus). A different algorithm, MLESAC (maximum likelihood sampling and consensus), was proposed in [124], where the cost function was modified to yield the maximum likelihood estimate under the assumption that the outliers are uniformly distributed. Like MSAC, MLESAC requires the parameters of the inlier noise model to be given by the user.

Various methods to improve on the random sampling step of RANSAC have also been proposed. In [120], the match scores from a point matching stage are used in the sampling. By replacing the random sampling in RANSAC with *guided sampling*, the probability of getting a *good* elemental subset was drastically improved. LO-RANSAC [22], enhances RANSAC with a local optimization step. This optimization executes a new sampling procedure based on how well the measurements satisfy the current best hypothesis. Alternatively, PROSAC [21] uses prior beliefs about the probability of a point being an inlier to modify the random sampling step of RANSAC. The robust estimation problem can also be treated as a Bayesian problem. This leads to a combination of RANSAC with importance sampling [121].

The original RANSAC algorithm is not applicable to real-time problems. In [88], a termination condition based on the execution time of the algorithm is used to limit sampling, so that RANSAC can be used for live structure from motion. For certain problems, the problem of degenerate data can be handled by detecting the degeneracy and using a different parametrization of the parameters being estimated [41].

In all this work, a major limitation to the practical use of RANSAC has not been addressed. RANSAC requires the user to specify the level of noise corrupting the inlier data. This estimate is known as the *scale* of the noise and the performance of RANSAC is sensitive to the accuracy of the scale estimate [77]. In many applications we have no knowledge of the true scale of inlier noise. The scale may also change with time and the system will have to adapt itself to these changes without user intervention. Consider a real-time image based reconstruction system which uses feature correspondences between frames to generate 3D models of the scene. Mismatches between frames will act as outliers during the estimation of the scene geometry and it will be necessary to use robust regression. The amount of noise corrupting the inliers will change based on how fast the camera is moving.

4.1.2 Robust Subspace Estimation

Given data lying in a N-dimensional space, linear regression estimates the one dimensional null space of the inliers. *Subspace estimation* requires the simultaneous estimation of k *linearly independent* vectors lying in the null space of the inliers. The intersection of the hyperplanes represented by these k constraints gives the required $N - k$ dimensional subspace containing all the inliers. Subspace estimation in the presence of outlier data is known as *robust subspace estimation.* If *all* the data points lie in the same subspace, then *Principal Component Analysis* (PCA) [63] would be sufficient, however PCA cannot handle outliers. Methods such as [3, 12] perform *robust PCA* to handle outliers, but these methods cannot handle structured outliers.

Subspace estimation occurs frequently in the analysis of dynamic scenes, where it is known as the *factorization problem* [27, 44, 118]. Since the seminal work of [118] which introduced factorization for orthographic cameras, factorization has been generalized to include affine cameras [94] and projective cameras [81, 125]. Many variations of the factorization algorithm have been proposed to handle difficulties such as multiple bodies [27, 44, 116, 135], outliers [66] and missing data [11, 54, 69]. The degeneracies are also well understood [117, 141]. We concentrate on factorization in the presence of outliers, both structured and unstructured. Structured outliers in factorization correspond to bodies with different motions and this is known as the *multibody factorization* problem [27, 44, 66, 96].

As we said in Section 3.6.1, factorization is based on the fact that if n rigidly moving points are tracked over F *affine* images, then the $2F$ image coordinates can be used to define feature vectors in $\mathbb{R}^{2F}$. These vectors lie in a four-dimensional subspace of $\mathbb{R}^{2F}$ [118]. If the data is centered then the dimension of the subspace is only three. Due to the large amount of research that is aimed at solving the factorization problem, we cannot offer a complete list of previous work done. Most work done in this area, including *subspace separation* [27, 66] and *generalized PCA* (GPCA) [135] aim to solve the multibody factorization problem where different bodies give rise to different subspaces.

Most methods make certain simplifying assumptions about the data. Firstly, in [27, 66] it is assumed that the subspaces are orthogonal. Therefore, for degenerate motions where the subspaces share a common basis vector, the methods break down [141]. Secondly, the methods of [27, 66] require the data to be centered, which is difficult to ensure in practice, especially in the presence of outliers. Finally, and most importantly, [27, 116, 135] do not account for unstructured outliers. For the purposes of estimating the subspace due to any particular motion, point tracks on other motions can be taken to be outliers. However, they do not consider *unstructured outliers*, *i.e.*, point tracks which do not lie on any of the bodies. Some preliminary work in this direction has been done in [134]. However, [134] assumes that even in the presence of outliers the algorithm returns a rough estimate of the true subspaces *and* the scale of the noise is known. None of these assumptions are guaranteed to hold in practice.

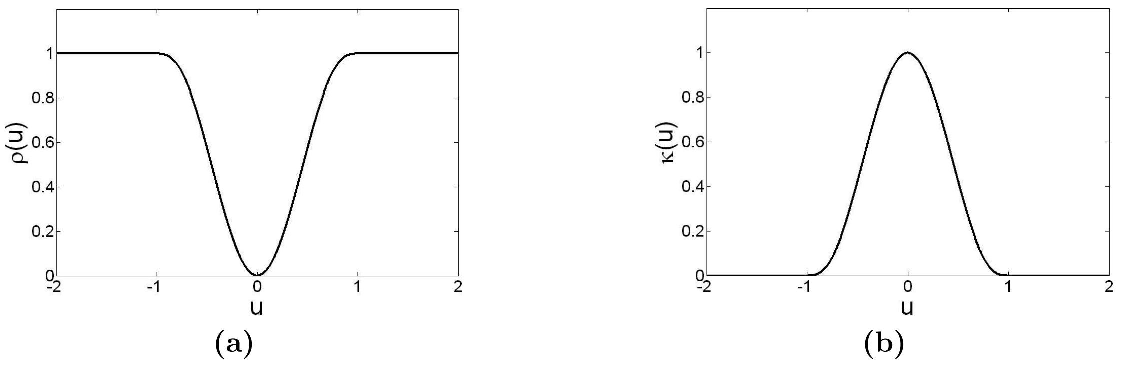


Figure 4.1: Biweight loss function, $\rho(u)$ and the biweight M-kernel function, $\kappa(u)$. Image (a) on the left is the *biweight* loss function and (b) is the corresponding *M-kernel* function.

4.1.3 Scale Independent Robust Regression

The pbM algorithm was initially proposed to solve the homoscedastic robust regression problem without requiring user defined scale estimates [14]. This was done by taking advantage of the relation between kernel density estimation and robust regression to propose data-driven scale selection rules.

The connection between nonparametric density estimation and robust regression has been remarked on before [25], and recently this equivalence has been used to develop scale independent solutions to the robust regression problem [104, 105, 138]. In [104] kernel density estimation is used to model the residuals of a regression based image model and to choose an appropriate bandwidth for mean shift based segmentation. The idea was extended in [105] to define a maximum likelihood estimator for parametric image models. In [138], a two step method was proposed for the regression problem. In the first step, they propose a robust scale estimator for the inlier noise. In the second step they optimize a cost function which uses the scale found in the first step. This requires the scale estimate to be close to the true scale. Kernel density estimation has also been used to propose novel scores such as the kernel power density [137].

The above methods were developed to handle homoscedastic noise for one-dimensional residuals. Robust subspace estimation involves multi-dimensional residuals and extending these methods is not trivial. For example, [138] uses a modified mean shift algorithm known as the mean-valley algorithm to find the minima of a kernel density along the real line. This method is unstable in one-dimension and will become worse in higher dimensional residual spaces which occur in subspace estimation.

In [110], the pbM algorithm was extended to handle heteroscedastic data. In the same work, and simultaneously in [99] for homoscedastic noise, a modified M-estimator cost function was introduced. The pbM algorithm was further extended to handle the robust subspace estimation problem in [113].

The pbM algorithm continues to use the hypothesise-and-test framework of the RANSAC family of algorithms. The sampling methods of the previously discussed algorithms can be used in the hypothesis generation part of pbM, while keeping the rest of the algorithm the same. Consequently, the advantages that pbM offers are different from methods such as PROSAC [21], QDEGSAC [41] etc. Our data-driven scale selection rules can be combined with any of the above algorithms to obtain improved robust regression methods.

4.2 Robust Heteroscedastic Linear Regression

The original pbM estimator was proposed as a solution to the robust linear errors-in-variables (EIV) problem [14]. It was later applied to the robust heteroscedastic-errors-in-variables (HEIV) problem which is more general then the linear EIV problem [110]. We begin by introducing the robust HEIV problem. We explain the role played by a user-defined scale estimate in RANSAC. We also connect the robust regression problem to kernel density estimation. This similarity is exploited by the pbM algorithm in Section 4.4.

Let $\mathbf{y}_{io} \in \mathbb{R}^p, \ i = 1, \ldots, n_1$ represent the true values of the inliers $\mathbf{y}_i$. Typically, for heteroscedastic regression, the data vectors are obtained from nonlinear mappings of lower dimensional image data. Given $n (> n_1)$ data points $\mathbf{y}_i, i = 1, \ldots, n$, we would like to estimate $\hat{\boldsymbol{\theta}} \in \mathbb{R}^p$ and $\hat{\alpha} \in \mathbb{R}$ such that the linear constraint,

$$\hat{\boldsymbol{\theta}}^T \hat{\mathbf{y}}_i - \hat{\alpha} = 0 \tag{4.1}$$

where,

$$\mathbf{y}_i = \mathbf{y}_{io} + \delta\mathbf{y}_i \qquad \delta\mathbf{y}_i \sim GI(0, \sigma^2\mathbf{C}_i)$$

for $i = 1, \ldots, n_1$. In the above equations, $\hat{\mathbf{y}}_i$ is an estimate of $\mathbf{y}_{io}$. The points $\mathbf{y}_i$, $i = n_1 + 1, \ldots, n$ are outliers and *no assumptions* are made about their distribution. The number of inliers, n_1, is unknown. The multiplicative ambiguity in (4.1) is removed by imposing the constraint $\|\boldsymbol{\theta}\| = 1$.

Note that each $\mathbf{y}_i$ $i = 1, \ldots, n_1$ is allowed to be corrupted by noise of a different covariance. This makes the problem *heteroscedastic* as opposed to homoscedastic, where all the covariances are the same. Heteroscedasticity usually occurs in vision due to nonlinear mappings between given image data and the data vectors involved in the regression. Given the covariance of the image data, the covariances of the regression data vectors can be found by error propagation [84]. For example, for fundamental matrix estimation, the lower dimensional image data vector is given by a vector in $\mathbb{R}^4$ which is mapped to a regression vector in $\mathbb{R}^8$. In this paper, we assume the covariance matrices, $\mathbf{C}_i$, are known up to a common scale [84].

The robust M-estimator formulation of this problem is

$$\left[\hat{\boldsymbol{\theta}}, \hat{\alpha}\right] = \arg\min_{\boldsymbol{\theta},\alpha} \frac{1}{n} \sum_{i=1}^{n} \rho\left(\frac{\boldsymbol{\theta}^T\mathbf{y}_i - \alpha}{s\sqrt{\boldsymbol{\theta}^T\mathbf{C}_i\boldsymbol{\theta}}}\right) \tag{4.2}$$

where, s is the *user-defined* scale parameter. The term, $\boldsymbol{\theta}^T\mathbf{y}_i - \alpha$ measures the deviation of the data from the required constraint. Deviations of points with larger covariances should have less weight than points with smaller covariances. This is achieved by the $\sqrt{\boldsymbol{\theta}^T\mathbf{C}_i\boldsymbol{\theta}}$ term which is the standard deviation of the projection, $\boldsymbol{\theta}^T\mathbf{y}_i$.

We use a *loss function*, $\rho(u)$, which is a redescending M-estimator. The loss function is non-negative, symmetric and non-decreasing with $|u|$. It has a unique minimum of $\rho(0) = 0$ and a maximum of one. Therefore, it penalizes points depending on how much they deviate from the constraint. Greater deviations are penalized more, with the maximum possible penalty being one. The scale, s, controls the level of error the cost function is allowed to tolerate. If the loss function is chosen to be the *zero-one* loss function

$$\rho_{0-1}(u) = \begin{cases} 0 & if \;\; |u| \leq 1 \\ 1 & if \;\; |u| > 1 \end{cases} \tag{4.3}$$

then (4.2) is equivalent to traditional RANSAC [37]. Some versions of RANSAC use continuous loss functions [124]. In our implementation, we use the *biweight* loss function (Figure 4.1a),

$$\rho(u) = \begin{cases} 1-(1-u^2)^3 & if \;\; |u| \leq 1 \\ 1 & if \;\; |u| > 1 \end{cases} \tag{4.4}$$

since it is continuous.

4.2.1 The Scale in RANSAC

The RANSAC algorithm solves the optimization problem of (4.2) by repeatedly *hypothesizing* parameter estimates and then *testing* them. An *elemental subset*, which is the minimum number of data points required to uniquely define a parameter estimate, is sampled randomly from the data and used to generate a parameter hypothesis $[\boldsymbol{\theta}, \alpha]$. For example, in the case of fundamental matrix estimation, we would use the eight-point algorithm to generate a fundamental matrix estimate. In the testing step, the robust score for this estimate is computed. The scale s used to compute the score is *given by the user*. This process is repeated till the system has attained a predefined confidence that a good hypothesis has been obtained [88]. The number of iterations depends on the fraction of inliers and the required confidence in the parameter estimates. The hypothesis with the best score is returned as the parameter estimate.

After the parameters have been estimated, the inliers in the data are separated from the outliers in the *inlier-outlier dichotomy* step. Let $\left[\hat{\boldsymbol{\theta}}, \hat{\alpha}\right]$ be the hypothesis with the best score. A data point $\mathbf{y}$ is declared an inlier if $\|\hat{\boldsymbol{\theta}}^T\mathbf{y} - \hat{\alpha}\| < s$, otherwise it is declared an outlier. The same user-defined scale s, is used in this step.

The scale estimate plays two important roles in RANSAC. Firstly, it appears in the robust score being optimized. Secondly, it is used to separate the inliers from the outliers in the final step. RANSAC requires the user to specify a single scale value to be used in both steps. Ideally, the value of scale used should be the true value, but for robust regression, choosing a good scale value is a circular problem. Given the scale value it is possible to estimate the parameters and the dichotomy using methods such as RANSAC. Given the inlier-outlier dichotomy, the parameters can be estimated using least squares regression and the scale can then be estimated through χ^2 tests.

In the absence of a good scale estimate it is not necessary for both the scale values to be the same. The reason RANSAC uses the value twice is that the scale estimate is defined by the user a single time. The pbM algorithm divides the robust estimation problem into two steps and uses two different *data-driven* scale values. One of these values appears in the robust score and this value is estimated by interpreting the robust score as a kernel density estimation. The inlier-outlier separation is based on a nonparametric analysis of the residual errors.

4.2.2 Weighted Kernel Density Estimation

Let $x_i, i = 1, \ldots, n$ be scalars sampled from an unknown distribution f. The *weighted adaptive kernel density estimate*

$$\hat{f}_K(x) = \frac{c_K}{nw} \sum_{i=1}^{n} \frac{w_i}{h_i} K\left(\frac{x - x_i}{h_i}\right) \tag{4.5}$$

based on a *kernel function*, K, and weights, w_i, satisfying

$$K(z) \geq 0 \qquad w_i \geq 0 \qquad w = \sum_{i=1}^{n} w_i \tag{4.6}$$

is a nonparametric estimator of the density $f(x)$ at x. The bandwidth h_i of x_i controls the width of the kernel placed at x_i and the weight w_i controls the importance given to the data point x_i. The constant c_k is chosen to ensure that $\hat{f}_K$ integrates to 1.

This is a slightly more general definition of a kernel density than the one given in Section 3.2. If all the weights are set to $w_i = 1$ and we use a symmetric kernel, $K(x) = k(x^2)$, where, $k(\cdot)$ is the *profile* then we obtain the previous definition of (3.1).

The mean shift vector for the weighted kernel density is obtained by taking the

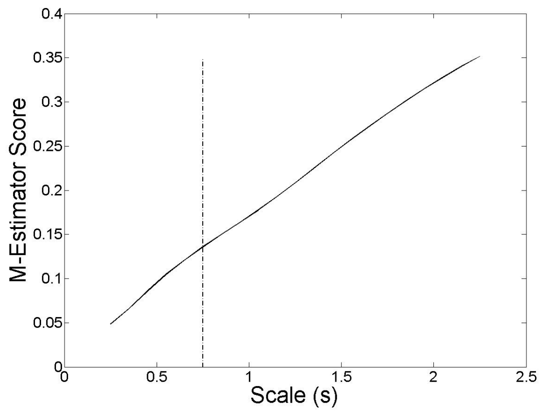


Figure 4.2: The quasi-linear variation of M-estimator scores with scale. The parameters $\boldsymbol{\theta}$ and α are set to their true values while the scale s is varied. The dashed vertical line indicates the true scale of the noise corrupting the inliers.

gradient of (4.5), and defining $g(z) = -k'(z)$,

$$m_h(x) = C\frac{\nabla \hat{f}_k(x)}{\hat{f}_g(x)} = \frac{\displaystyle\sum_{i=1}^{n} \frac{w_i x_i}{h_i^3} g\left(\left\|\frac{x - x_i}{h_i}\right\|^2\right)}{\displaystyle\sum_{i=1}^{n} \frac{w_i}{h_i^3} g\left(\left\|\frac{x - x_i}{h_i}\right\|^2\right)} - x. \tag{4.7}$$

C is a positive constant and $m_h(x)$ is the *mean shift* vector. The mean shift is proportional to the normalized density gradient and the iteration

$$x^{(j+1)} = m_h(x^{(j)}) + x^{(j)} \tag{4.8}$$

is a gradient ascent technique converging to a stationary point of the kernel density. Saddle points are detected and removed, to obtain the modes of $\hat{f}_K(x)$.

4.2.3 Reformulating the Robust Score

The heteroscedastic M-estimator formulation of (4.2) is mathematically similar to the kernel density estimation of (4.5). To make this similarity precise, we replace the loss function $\rho(u)$ in (4.2) by the *M-kernel function* $\kappa(u) = 1-\rho(u)$. The robust M-estimator problem now becomes

$$\left[\hat{\boldsymbol{\theta}}, \hat{\alpha}\right] = \arg\max_{\boldsymbol{\theta},\alpha} \frac{1}{n}\sum_{i=1}^{n} \kappa\left(\frac{\boldsymbol{\theta}^T \mathbf{y}_i - \alpha}{s\sqrt{\boldsymbol{\theta}^T \mathbf{C}_i \boldsymbol{\theta}}}\right). \tag{4.9}$$

The M-kernel function corresponding to the biweight loss function is given by (Figure 4.1b)

$$\kappa(u) = \begin{cases} (1-u^2)^3 & if \;\; |u| \leq 1 \\ 0 & if \;\; |u| > 1. \end{cases} \tag{4.10}$$

Now, suppose we fix the direction $\boldsymbol{\theta}$. The projections of the data points $\mathbf{y}_i$ along this direction are given by $\boldsymbol{\theta}^T\mathbf{y}_i$ and the covariance of this projection is given by $\boldsymbol{\theta}^T\mathbf{C}_i\boldsymbol{\theta}$. The robust score of (4.9) can be thought of as an *adaptive kernel density estimate* with the one-dimensional data points being the projections $\boldsymbol{\theta}^T\mathbf{y}_i$. The mode of this density will be the intercept α. We choose the kernel function K to be the M-kernel function κ and the bandwidths h_i and weights w_i are chosen appropriately, as shown in Table 4.1 below.

Table 4.1: Kernel Density Estimation and M-Estimators

	KDE	M-Estimators
Kernel	K	κ
Bandwidth	h_i	$s\sqrt{\boldsymbol{\theta}^T\mathbf{C}_i\boldsymbol{\theta}}$
Weights	w_i	$\sqrt{\boldsymbol{\theta}^T\mathbf{C}_i\boldsymbol{\theta}}$

The kernel density (4.5) of the projections along the direction $\boldsymbol{\theta}$ becomes

$$\hat{f}_{\boldsymbol{\theta}}(x) = \frac{c_\kappa}{nsw}\sum_{i=1}^{n}\kappa\left(\frac{\boldsymbol{\theta}^T\mathbf{y}_i - x}{s\sqrt{\boldsymbol{\theta}^T\mathbf{C}_i\boldsymbol{\theta}}}\right). \tag{4.11}$$

The factor c_κ is a constant and can be ignored. The kernel density equation of (4.11) differs from the M-kernel formulation of (4.9) only by a division by s and w.

The term w appears in the kernel density to ensure that it is in fact a density which integrates to one. It is the sum of terms of the form $\sqrt{\boldsymbol{\theta}^T\mathbf{C}_i\boldsymbol{\theta}}$ which depend on the covariances $\mathbf{C}_i$, which are part of the data, and the direction $\boldsymbol{\theta}$. The aim of the robust regression algorithm is to maximize a robust M-score and kernel density estimation is used as a computational tool to achieve this. Since w is not part of the robust M-score we do not include it in the cost function being optimized. For a constant $\boldsymbol{\theta}$, w does not change and acts as a proportionality factor of the kernel density which does not affect the position of the maximum of the density.

The data driven scale is depends on the projections of the data points and therefore varies as the direction of projection $\boldsymbol{\theta}$ changes. When comparing M-estimator scores for different directions, we are comparing scores evaluated at different scales. It is necessary to account for the variation of the M-estimator score due to the change in the scale. In Figure 4.2 the M-estimator score is computed at different scale values for a randomly generated data set of 40 inliers lying on a plane and 40 outliers. The values of $\boldsymbol{\theta}$ and α are set to their true values and the dashed vertical line shows the true scale which is 0.75. It can be seen that as the scale changes the score variation is almost linear.

In previous hypothesise-and-test algorithms, the scale was held constant and did not affect the optimization. However, as s increases, the M-score of (4.9) increases quasi-linearly with s. Using data driven scale selection rules would bias the system in favor of more distributed residuals unless we account for the variation of the M-score with scale. To do this, we maximize the ratio between the M-score and the scale at which it is evaluated. Denoting the data driven scale by $s_{\boldsymbol{\theta}}$, the M-estimator formulation becomes

$$\left[\hat{\boldsymbol{\theta}}, \hat{\alpha}\right] \quad = \quad \arg\max_{\boldsymbol{\theta},\alpha} \frac{1}{n s_{\boldsymbol{\theta}}} \sum_{i=1}^{n} \kappa \left(\frac{\boldsymbol{\theta}^T \mathbf{y}_i - \alpha}{s_{\boldsymbol{\theta}} \sqrt{\boldsymbol{\theta}^T \mathbf{C}_i \boldsymbol{\theta}}} \right) \tag{4.12}$$

and the corresponding kernel density formulation reads

$$\left[\hat{\boldsymbol{\theta}}, \hat{\alpha}\right] \quad = \quad \arg\max_{\boldsymbol{\theta},\alpha} \hat{f}_{\boldsymbol{\theta}}(\alpha). \tag{4.13}$$

The cost function (4.12) does not integrate to one and is no longer a kernel density. However, it is proportional to a kernel density estimate and the position of the maximum does not change. Using mean shift to find the mode of the projections while holding $\boldsymbol{\theta}$ constant would still be valid.

In our formulation, we return the hypothesis with the highest *ratio* between the M-score and the scale at which it is evaluated. RANSAC, on the other hand, holds the scale constant and returns the hypothesis with the highest M-score. A similar cost function was proposed in [99], based on the empirical performance of pbM for homoscedastic data. However, the theoretical justification for the division by scale due to the linear variation of the M-score was first pointed out in [110].

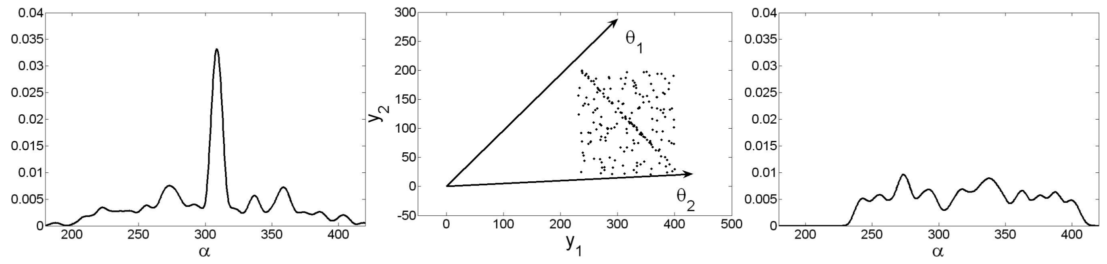


Figure 4.3: An example of projection pursuit. The 2D data points and the two directions, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are shown in the middle image. The kernel density estimate of the projections along $\boldsymbol{\theta}_1$ is shown on the left. There is a clear peak at the intercept. The projections along $\boldsymbol{\theta}_2$ give a more diffuse density, as seen in the right figure.

4.2.4 Projection Pursuit

The kernel density formulation offers an alternate justification for the new robust score of (4.12). Given a direction, the intercept is the largest mode of the kernel density. The direction with the highest density at the mode is the estimated direction. This approach to the robust heteroscedastic errors-in-variables problem is known as *projection pursuit* in statistics. The equation (4.13) can be rewritten as

$$\left[\hat{\boldsymbol{\theta}}, \hat{\alpha}\right] = \arg\max_{\boldsymbol{\theta}} \left[\max_{x} \hat{f}_{\boldsymbol{\theta}}(x)\right] . \tag{4.14}$$

The inner maximization on the right hand side returns the intercept α as a function of $\boldsymbol{\theta}$ and this is the *projection index* of $\boldsymbol{\theta}$.

$$\alpha = \arg\max_{x} \hat{f}_{\boldsymbol{\theta}}(x). \tag{4.15}$$

The direction with the maximum projection index is the robust estimate.

The projection pursuit approach towards M-estimation has a clear geometric interpretation. The direction $\boldsymbol{\theta}$ can be regarded as the unit normal of a candidate hyperplane fitted to the p-dimensional data. The bandwidth $s_{\boldsymbol{\theta}}$ defines a band along this direction. The band is translated in $\mathbb{R}^p$, along $\boldsymbol{\theta}$, to maximize the M-score of the orthogonal distances from the hyperplane. The M-estimate corresponds to the densest band over all $\boldsymbol{\theta}$.

These ideas are geometrically illustrated in Figure 4.3, for two-dimensional data. The inliers, which lie close to a line, and the outliers, which are uniformly distributed, are shown in the middle figure. Their projections are taken along two directions, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. The kernel density estimate of the projections along $\boldsymbol{\theta}_1$ is shown on the left and it exhibits a clear mode at the intercept. The kernel density estimate based on the projections along $\boldsymbol{\theta}_2$ is more diffuse. The mode is not that high and consequently, $\boldsymbol{\theta}_2$ will have a much lower projection index than $\boldsymbol{\theta}_1$.

4.3 Robust Subspace Estimation

In the last section we reformulated the robust regression problem as a projection pursuit problem with the projection score given by a kernel density. We can carry out a similar procedure to rewrite the robust subspace estimation problem as a projection pursuit problem [113]. In this section we start from the robust M-estimator formulation and derive the equivalent M-kernel and kernel density forms.

Let $\mathbf{y}_{io}$, $i = 1, \ldots, n_1$, be the true value of the inlier data points $\mathbf{y}_i$. Given $n(> n_1)$ data points $\mathbf{y}_i$, $i = 1, \ldots, n$, the problem of subspace estimation is to estimate $\boldsymbol{\Theta} \in \mathbb{R}^{N\times k}$, $\boldsymbol{\alpha} \in \mathbb{R}^k$

$$\boldsymbol{\Theta}^T \mathbf{y}_{io} - \boldsymbol{\alpha} = \mathbf{0}_k \tag{4.16}$$

where

$$\mathbf{y}_i = \mathbf{y}_{io} + \delta\mathbf{y}_i \qquad \delta\mathbf{y}_i \sim GI(0, \sigma^2 \mathbf{I}_{N\times N})$$

and, σ is the *unknown* scale of the noise. Handling non-identity covariances for heteroscedastic data, is a simple extension of this problem, *e.g.*, [83]. The multiplicative ambiguity is resolved by requiring $\boldsymbol{\Theta}^T\boldsymbol{\Theta} = \mathbf{I}_{k\times k}$.

Given a set of k linearly independent constraints, they can be expressed by an equivalent set of *orthonormal* constraints. The $N \times k$ orthonormal matrix $\boldsymbol{\Theta}$ represents the k constraints satisfied by the inliers. The inliers have $N - k$ degrees of freedom and lie in a subspace of dimension $N - k$. Geometrically, $\boldsymbol{\Theta}$ is the basis of the k dimensional null space of the data and is a point on the Grassmann manifold, $G_{N,k}$ [31]. We assume k is known and here we do not treat the case where the data may be degenerate and lie in a subspace of dimension less than k. Usually, $\boldsymbol{\alpha}$ is taken to be zero since any subspace must contain the origin. However, for a robust problem, where the data is not centered, $\boldsymbol{\alpha}$ represents an estimate of the centroid of the inliers. Since we are trying to estimate both $\boldsymbol{\Theta}$ and $\boldsymbol{\alpha}$, the complete search space for the parameters is $G_{N,k} \times \mathbb{R}^k$. The projection of $\boldsymbol{\alpha}$ onto the column space of $\boldsymbol{\Theta}$ is given by $\boldsymbol{\Theta\alpha}$. If we use a different basis to represent the subspace, $\boldsymbol{\alpha}$ will change such that the product $\boldsymbol{\theta\alpha}$ is constant.

The scale from the one-dimensional case now becomes a scale matrix and to account for the variation of the M-score with scale, we now have to divide by the determinant of the scale matrix. The robust M-estimator version of the subspace estimation problem is

$$\left[\hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{\alpha}}\right] = \arg\min_{\boldsymbol{\theta},\boldsymbol{\alpha}} \frac{1}{n\left|\mathbf{S}_{\boldsymbol{\Theta}}\right|^{1/2}} \sum_{i=1}^{n} \rho\left(\mathbf{x}_i^T \mathbf{S}_{\boldsymbol{\Theta}}^{-1} \mathbf{x}_i\right) \tag{4.17}$$

where, $\mathbf{x}_i = \boldsymbol{\Theta}^T \mathbf{y}_i - \boldsymbol{\alpha}$, $\mathbf{S}_{\boldsymbol{\Theta}}$ is a scale matrix and $\left|\mathbf{S}_{\boldsymbol{\Theta}}\right|$ is its determinant. The function $\rho(u)$ is the biweight loss function of (4.4). The M-estimator problem can be rewritten in terms of the M-kernel function $\kappa(u)$ as

$$\left[\hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{\alpha}}\right] = \arg\max_{\boldsymbol{\Theta},\boldsymbol{\alpha}} \frac{1}{n\left|\mathbf{S}_{\boldsymbol{\Theta}}\right|^{1/2}} \sum_{i=1}^{n} \kappa\left(\mathbf{x}_i^T \mathbf{S}_{\boldsymbol{\Theta}}^{-1} \mathbf{x}_i\right). \tag{4.18}$$

Building the same analogies as in Section 4.2.3, the M-kernel formulation of (4.18) can be shown to be equivalent to kernel density estimation in $\mathbb{R}^k$

$$\hat{f}_{\boldsymbol{\Theta}}(\mathbf{x}) = \frac{1}{n\left|\mathbf{S}_{\boldsymbol{\Theta}}\right|^{1/2}} \sum_{i=1}^{n} \kappa\left(\mathbf{x}_i^T \mathbf{S}_{\boldsymbol{\Theta}}^{-1} \mathbf{x}_i\right) \tag{4.19}$$

with bandwidth $\mathbf{S}_{\boldsymbol{\Theta}}$ and kernel $\kappa(u)$. The robust M-estimator problem of (4.17) can be stated as

$$\left[\hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{\alpha}}\right] = \arg\max_{\boldsymbol{\Theta}} \left[\max_{\mathbf{x}} \hat{f}_{\boldsymbol{\Theta}}(\mathbf{x})\right]. \tag{4.20}$$

This is a projection pursuit definition of the problem, and the inner maximization returns the intercept as function of $\boldsymbol{\Theta}$

$$\boldsymbol{\alpha} = \arg\max_{\mathbf{x}} \hat{f}_{\boldsymbol{\Theta}}(\mathbf{x}). \tag{4.21}$$

This maximization can be carried out by mean shift in $\mathbb{R}^k$ [25].

4.4 The Projection Based M-estimator

We now develop the projection based M-estimator (pbM) algorithm to handle the robust subspace segmentation problem of (4.20). The robust regression problem of (4.14) is a special case of this with $k = 1$.

The pbM algorithm begins by sampling elemental subsets, without replacement, from the given data set. An elemental subset is used to get an initial estimate $\boldsymbol{\Theta}$. Given $\boldsymbol{\Theta}$, the intercept $\boldsymbol{\alpha}$ is estimated according to (4.21). This mode search is done through mean shift [25]. To perform mean shift it is necessary to choose a scale, and we propose a data driven scale selection rule for this purpose in Section 4.4.1. The density at $\boldsymbol{\alpha}$ is given by (4.19) with $\mathbf{x}_i = \boldsymbol{\Theta}^T \mathbf{y}_i - \boldsymbol{\alpha}$. Recall that in RANSAC, both the direction $\boldsymbol{\Theta}$ *and* the intercept $\boldsymbol{\alpha}$ are generated by sampling. In our case, only the choice of $\boldsymbol{\Theta}$ depends on the elemental subset while the intercept depends on the projections of *all* the measurement data.

This sampling step is repeated a number of times. After each sampling step, given a parameter hypothesis, we perform local optimization to improve the score in a neighbourhood of the current parameter estimates. The idea this is that given a $[\boldsymbol{\Theta}, \boldsymbol{\alpha}]$ pair, the robust score can be improved by running a local search in the parameter space. Local optimization typically improves the score marginally and thus it is not necessary to perform the optimization for every elemental subset. We set a threshold $0 < \gamma < 1$, and the local optimization is performed only when the current score is greater than γ times the highest score obtained so far. The local optimization procedure is discussed in Section 4.4.2.

The parameter pair with the highest score is returned as the robust parameter estimate $[\hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{\alpha}}]$. Given $[\hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{\alpha}}]$, the inlier/outlier dichotomy estimation is also completely data-driven. We discuss the inlier-outlier separation procedure in Section 4.4.3.

4.4.1 Data-driven Scale Selection

The formulation of the M-estimator score as a kernel density offers a computational advantage. If $\boldsymbol{\Theta}$ is close to the true value of the model, it is sufficient to choose a

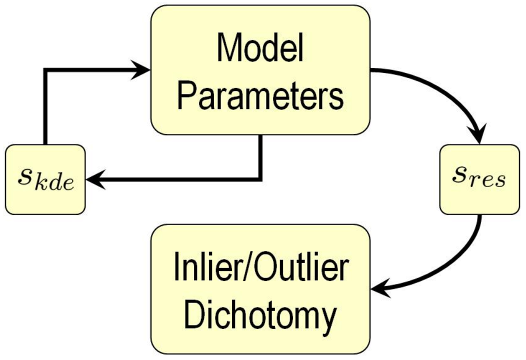


Figure 4.4: The pbM algorithm uses two different scales. Data-driven scales are used for computing a robust M-scores. A new scale is then used to separate the inlier from the outliers.

scale which ensures that the maximization (4.21) returns good estimate of the true intercept. This is a much easier condition to satisfy than requiring the scale to be a good estimate of the *unknown* noise scale. Furthermore, for kernel density estimation, there exist plug in rules for bandwidth selection which are *purely data driven.* In [136, Sec.3.2.2] the following bandwidth selection rule was proposed which was shown to give good asymptotic convergence properties for one-dimensional kernel density estimation

$$s = n^{-1/5} \operatorname*{med}_j \left| x_j - \operatorname*{med}_i x_i \right| . \tag{4.22}$$

where $x_i, i = 1, \ldots, n$ are the data points being used to define the kernel density.

For subspace estimation, we have k-dimensional residuals of the form $\boldsymbol{\Theta}^T \mathbf{y}_i$. We extend the one-dimensional rule (4.22) by applying it k times, once for each component of the residuals. This gives k different scale values. The bandwidth matrix, $\mathbf{S}_{\boldsymbol{\Theta}}$, is a *diagonal* matrix with the value at (j, j) given by the square of the scale of the j-th residual. For a different $\boldsymbol{\Theta}$, the residuals are different and we get a different bandwidth matrix. This variation with $\boldsymbol{\Theta}$ is the reason for dividing by the term $\left|\mathbf{S}_{\boldsymbol{\Theta}}\right|^{1/2}$ in (4.18).

For the robust regression case with $k = 1$, the scale selection rule is applied to one-dimensional residuals $\boldsymbol{\theta}^T \mathbf{y}_i$. For the robust subspace estimation problem with $k > 1$, the bandwidth matrix depends on the basis used to represent the null space. If we use a different basis to represent the same subspace, the residuals are transformed by some rotation matrix. Ideally, we would like the bandwidth matrix to also undergo an equivalent rotation but this is not so. However, we only require the bandwidth matrix to be good enough to get a good estimate of the mode of the kernel density estimate. For this purpose, choosing the scale along each direction is sufficient.

Replacing the scale by the bandwidth matrix of the kernel density estimate is not a simple substitution. For M-estimators and RANSAC, the scale appears in the cost function and points with errors greater than the scale are rejected as outliers. For pbM, the bandwidth is data driven and is only used in estimating the density of projected points to find the mode. It is *not* a threshold for acceptance of inliers (Figure 4.4).

4.4.2 Local Optimization

The robust scores defined in (4.14) and (4.20) are non-differentiable. This is due to the complex dependence of the bandwidths and the intercepts on the basis $\mathbf{\Theta}$, or $\boldsymbol{\theta}$ in the single constraint case. The bandwidth depends on $\mathbf{\Theta}$ through (4.22) and this function is clearly not differentiable. The intercept $\boldsymbol{\alpha}$ also depends on $\mathbf{\Theta}$ through a non-differentiable function. For example, consider a data set consisting of two different structures which satisfy the projection constraints with the parameter values $[\boldsymbol{\alpha}_1, \mathbf{\Theta}_1]$ and $[\boldsymbol{\alpha}_2, \mathbf{\Theta}_2]$. Initially we set the projection direction to $\mathbf{\Theta}_1$, and the maximization of (4.21) would return the value $\boldsymbol{\alpha}_1$. Now as we move the projection direction from $\mathbf{\Theta}_1$ to $\mathbf{\Theta}_2$ the position of the mode would move smoothly till at some point both the structures give rise to modes of equal height. On moving the direction some more, there will be a jump in the estimate returned by the maximization of (4.21). This sudden change makes the problem (4.20) discontinuous and hence, non-differentiable.

In [14], simplex optimization was used for the homoscedastic single constraint case, since this does not require the computation of gradients. However, simplex optimization requires a minimal parametrization of the search space. For $\mathbf{G}_{N,k}, k > 1$ such parametrizations are very difficult to work with.

We use derivative based methods for the local search. This is done by approximating the robust score so that its gradient can be computed. The bandwidth matrix is assumed to be constant and its dependence on $\mathbf{\Theta}$ is ignored. The intercept is treated as an independent variable and the local search is done over the product space, $\mathbf{G}_{N,k} \times \mathbb{R}^k$. The conjugate gradient algorithm over $\mathbf{G}_{N,k} \times \mathbb{R}^k$ is presented in Appendix A. The single constraint case is a special case of this with $k = 1$.

Previous work on adding local optimization to RANSAC has proposed discrete optimization techniques [22]. This method modifies the probability of a point being an inlier based on how well is satisfies a parameter hypothesis and perform an inner sampling with the modified probabilities. Our optimization uses continuous optimization of the robust score in the parameter space over which we are searching. Another proposal is to explicitly trade-off local exploration of the parameter space with exploitation or sampling [50].

In our experiments we find that even without the local optimization step, using the cost function proposed here to get robust estimates gives better results than RANSAC. However, the local optimization step further improves results and makes the estimates less sensitive to the additive noise corrupting the inliers. The parameter γ controls the number of times the local optimization is performed. Setting $\gamma = 0$ leads to the local optimization being performed for each elemental subset. This leads to the algorithm of [113]. Although it is reasonable to use values of $\gamma < 1$, theoretically we can use any positive value for γ. Setting γ to a large positive value ensures the optimization is never performed. Local optimization is the most computationally expensive part of the algorithm, but using a threshold of $\gamma = 0.9$ gives good results while not affecting the complexity of the algorithm much.

4.4.3 Inlier-Outlier Dichotomy Generation

The inlier-outlier separation is done only once after repeating the random sampling a sufficient number of times (Figure 4.4). The inlier-outlier dichotomy algorithm is based on the assumption of unimodal additive noise corrupting the inliers. Given the parameters with the highest robust score, $[\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\Theta}}]$, all the data points are projected to $\mathbb{R}^k$ as $\hat{\boldsymbol{\Theta}}^T \mathbf{y}_i$. If, $\hat{\boldsymbol{\Theta}}$ is close to the true direction, the projections of the inliers should form a single mode at the intercept $\hat{\boldsymbol{\alpha}}$. We estimate the kernel density of this mode and only points lying in the *basin of attraction* of this mode are declared inliers. A point lies in the basin of attraction if the mean shift iterations initialized at that point converge to $\hat{\boldsymbol{\alpha}}$ with some tolerance. Points which do not lie in the basin of attraction are declared outliers. If $k = 1$, the basin of attraction of the mode corresponds exactly to the window between the two minima on either side of the mode at $\hat{\alpha} \in \mathbb{R}$ [14, 110].

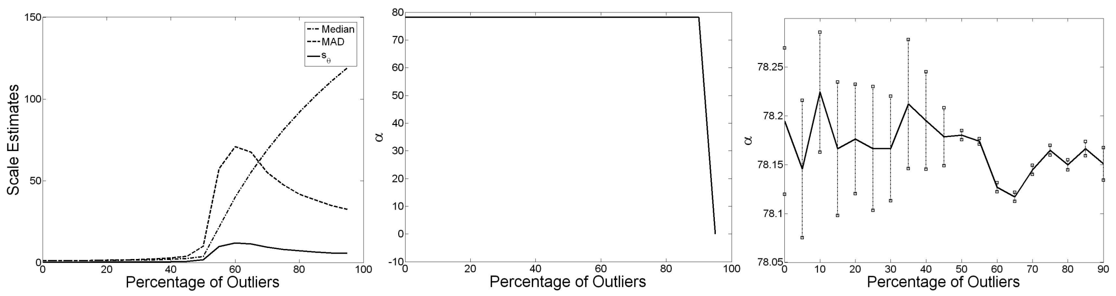
If the data lies along two parallel hyperplanes, then in the first estimation the kernel density estimate of the residuals will show two strong modes. The higher mode is retained as the intercept. The points on the other hyperplane are classified as outliers. By running pbM on the point classified as outliers from the first step, the second structure can be estimated. The complete algorithm is summarized below.

```
Algorithm: PROJECTION BASED M-ESTIMATOR

Given: Data points $\mathbf{y}_i, i = 1, \ldots, n$, $\gamma$, $f_{max} = 0$

    for $j \leftarrow 1 \ldots m$
        Sample elemental subsets and estimate $\boldsymbol{\Theta}_j$.
        Estimate scale matrix $S_{\boldsymbol{\Theta}_j}$.
        Do mean shift with scale $S_{\boldsymbol{\Theta}_j}$ to get $\boldsymbol{\alpha}_j$.
        if $\hat{f}_{\boldsymbol{\Theta}_j}(\boldsymbol{\alpha}_j) > \gamma f_{max}$
            Do local search to improve $\hat{f}_{\boldsymbol{\Theta}_j}(\boldsymbol{\alpha}_j)$.
        if $\hat{f}_{\boldsymbol{\Theta}_j}(\boldsymbol{\alpha}_j) > f_{max}$
            $f_{max} = \hat{f}_{\boldsymbol{\Theta}_j}(\boldsymbol{\alpha}_j)$
            $\left[\hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{\alpha}}\right] = [\boldsymbol{\Theta}_j, \boldsymbol{\alpha}_j]$

    Perform inlier-outlier dichotomy with $\left[\hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{\alpha}}\right]$.
    Return $\left[\hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{\alpha}}\right]$ and inliers.
```



(a) (b) (c)

Figure 4.5: Scale selection experiment with synthetic data with $k = 1$. Figure (a) compares the various scale estimators' performance as the number of outliers increase. Figure (b) shows the mode estimate computed on the same data sets and (c) shows a zoomed-in version of (b).

4.5 Results

We present the results of pbM and compare it to other robust estimators for two real applications. Affine motion estimation and fundamental matrix estimation requires the use of the heteroscedastic version of pbM, while affine factorization requires multiple linear constraints to be enforced simultaneously. Other applications can be found in our previous work [14, 110, 113].

4.5.1 Synthetic Data

As mentioned in Section 4.2, the major advantage of pbM is the fact that it decouples the scale selection problems at the parameter estimation and dichotomy generation stages. Here we verify this claim for the parameter estimation step with $k = 1$. We only require the scale to be good enough to accurately find the mode of the kernel density of (4.12) while holding $\boldsymbol{\theta}$ constant.

Synthetic data lying on a plane in $\mathbb{R}^3$ and corrupt it with additive Gaussian noise of standard deviation one was generated. The outliers are distributed uniformly in a cube centered around the origin with each side of length 200. As the number of outliers is increased, different scale selection rules are compared. For this experiment only, we assume the true parameters are known. This is because all scale selection rules assume a parameter estimate is available and estimate the scale based on the residuals. We use the true parameter values to compute the residuals. In practice, the situation can only get worse since the true parameters are unknown.

We consider the performance of various scale estimators as the fraction of inliers changes and compare their performances in Figure 4.5. We compare the median scale estimate

$$s_{med} = \underset{i}{\operatorname{med}} \left| \boldsymbol{\theta}^T \mathbf{y}_i - \alpha \right| \tag{4.23}$$

the Median Absolute Deviations (MAD) scale estimate

$$s_{mad} = \underset{j}{\operatorname{med}} \left| \boldsymbol{\theta}^T \mathbf{y}_j - \underset{i}{\operatorname{med}} \boldsymbol{\theta}^T \mathbf{y}_i \right| \tag{4.24}$$

and the plug-in rule of (4.22) with $x_i = \boldsymbol{\theta}^T \mathbf{y}_i$. The MAD scale we use is the same as the usual MAD rule with a scaling constant $c = 1.0$. The only difference between the MAD scale and the plug-in rule is the extra factor of $n^{-1/5}$ where n is the number of data points. Note, that while MAD and the plug-in rule do not require the data to be centered, (*i.e.*, only $\boldsymbol{\theta}$ is required, not α), the median scale requires the centered residuals.

The total number of data points is held constant at 1000 and the number of inliers and outliers is changed. For each value of the fraction of outliers, 100 data sets were generated and the scale estimators were used to get the inlier scale. The average value of the 100 trials is plotted as a function of the fraction of outliers in Figure 4.5a. As the number of outlier increases, the performance of all the scale selection rules suffers. As expected, the breakdown point of all three estimators is at most 50%. The standard deviations of the estimates across the 100 trials are very low for fewer outliers but beyond fifty percent, these standard deviations increase rapidly.

For each of these data sets we use mean shift to find the intercept. The scale used in the mean shift is given by (4.22). The average intercept estimate of 100 trials as a function of the percentage of outliers is shown in Figure 4.5b. The true value of the intercept is 78.2. This estimator is extremely accurate, and it only breaks down when over 90% of the data is outlier data. In Figure 4.5c we show a zoomed in version of the curve. The dashed lines indicate the variance by showing points which are one standard

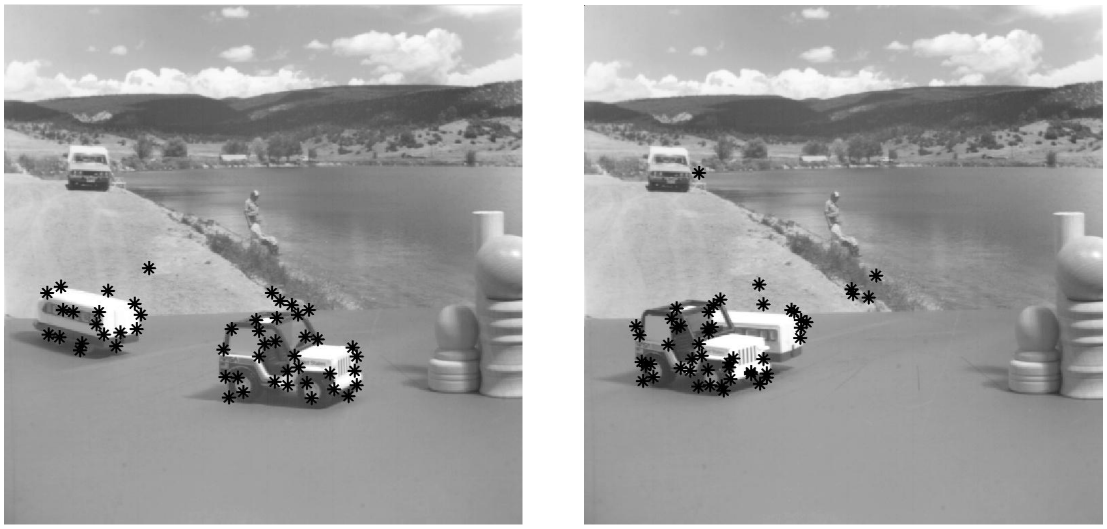


Figure 4.6: Images used for affine motion estimation. All the 51 points (inliers and outliers) matched in the two views are shown.

deviation away. When thought of as an estimator of the intercept, given the direction, mode finding is clearly robust and accurate. In fact, the variance of the mode estimate decreases as the number of outliers increases since the total number of points is held constant. As the outliers increase the number of inliers decreases and define a mode with lower variance.

4.5.2 Affine Motion Estimation

For 2D affine transformation associated with a moving object, $\mathbf{y}_{io} = [y_{i1o} \;\; y_{i2o}]$, $i = 1, 2$, are the (unknown) true coordinates of a pair of salient points in correspondence. The six parameter affine transformation between them

$$\begin{bmatrix} y_{21o} \\ y_{22o} \end{bmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{bmatrix} y_{11o} \\ y_{12o} \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \tag{4.25}$$

can be decoupled into two three-dimensional problems, in a_{11}, a_{12}, t_1 and a_{21}, a_{22}, t_2 respectively, each obeying a linear model. Thus, the noisy measurements of corresponding points are distributed around two planes in two different 3D spaces. The transformation parameters can be found through two separate estimation processes, and points obeying the transformation must be inliers for *both* processes.

The images used in the experiment are shown in Figure 4.6. A large number of point

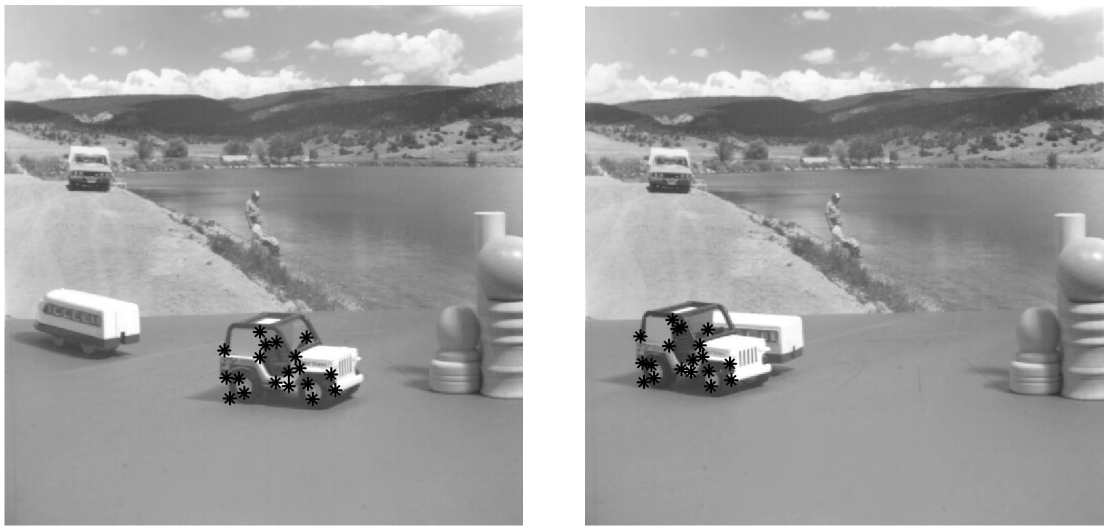


Figure 4.7: Results of affine motion estimation. The 19 inliers returned by pbM are shown in the two images.

correspondences were established using [47]. Next, the point matches on the static background were identified by having zero displacement and removed. The estimation process used the remaining 51 point correspondences of which 21 are inliers. The covariances of the matched points are found using the method of [67] and using a 7×7 window around each corner. Each data point has a different covariance and the data is heteroscedastic.

The performance of pbM is compared with RANSAC and MSAC in Table 4.2. For the ground truth, the inliers were manually selected. This was used to estimate the inlier noise standard deviation for each of the two three-dimensional problems, $\sigma_t^{(1)} = 1.62$, $\sigma_t^{(2)} = 1.17$. The final RANSAC and MSAC estimates were obtained by applying them to the two subproblems and then combining the results. In a real application the ground truth is unknown and the scale of the inlier noise is also unknown. To simulate the real situation, RANSAC and MSAC were also run after tuning them to the MAD scale estimate given by (4.24). In all our experiments, the same identical elemental subsets were used by all the algorithms to ensure that sampling does not bias any one algorithm. Performance is compared based on the number of true inliers among the points classified as inliers by the estimator. We also compare the estimators based on the ratio between the noise standard deviation of the selected points and the standard deviation of the inlier noise. The closer the measures are to unity the better the performance. The inliers found by pbM are shown in Figure 4.7.

Table 4.2: Performance Comparison - Affine Motion Estimation

	sel./in.	$\sigma_{in}^{(1)}/\sigma_t^{(1)}$	$\sigma_{in}^{(2)}/\sigma_t^{(2)}$
RANSAC(σ_{opt})	9/9	0.30	0.28
MSAC(σ_{opt})	9/9	0.29	0.28
RANSAC(σ_{mad})	14/10	9.20	22.83
MSAC(σ_{mad})	12/10	9.76	16.52
pbM	19/19	1.13	0.92

RANSAC and MSAC give similar performances when the true scale of noise is known. The performance degrades when the MAD scale estimate is used while pbM does better then both of them even without any scale estimate.

4.5.3 Fundamental Matrix Estimation

For fundamental matrix estimation it is necessary to account for the heteroscedasticity of the data [83]. The fundamental matrix between two images of the same scene expresses the geometric *epipolar* constraint on corresponding points. The constraint is bilinear, and can be expressed as $[\mathbf{x}_{1o}^T \;\; 1]\mathbf{F}[\mathbf{x}_{2o}^T \;\; 1]^T = 0$ where the matrix $\mathbf{F}$ has rank 2. On linearizing we get

$$\hat{\mathbf{y}} = [\hat{\mathbf{x}}_1^T \;\; \hat{\mathbf{x}}_2^T \;\; \mathbf{vec}(\hat{\mathbf{x}}_1\hat{\mathbf{x}}_2^T)]^T \in \mathbb{R}^8 \tag{4.26}$$

$$[\hat{\boldsymbol{\theta}}^T \;\; \hat{\alpha}]^T = \mathbf{vec}(\hat{\mathbf{F}}) \tag{4.27}$$

$$\hat{\boldsymbol{\theta}}^T\hat{\mathbf{y}} - \hat{\alpha} = 0 \qquad \|\hat{\boldsymbol{\theta}}\| = 1 \tag{4.28}$$

where the **vec** operator transforms a matrix into its column-organized form. The parameter vector $\boldsymbol{\theta}$ corresponds to the elements F_1 to F_8 and α corresponds to F_9 up to scale. In the absence of any further knowledge it is reasonable to assume that the given estimates of $\mathbf{x}_{1o}$ and $\mathbf{x}_{2o}$ are corrupted by homoscedastic normal noise with identity covariance. However, the linearized data vectors $\mathbf{y}$ are bilinear functions of the point locations $\mathbf{x}_1$ and $\mathbf{x}_2$, and therefore the vectors $\mathbf{y}$ do *not* have the same covariances. The data vectors for the regression are heteroscedastic [83]. The linearized data vector covariances are found by error propagation similar to [71].

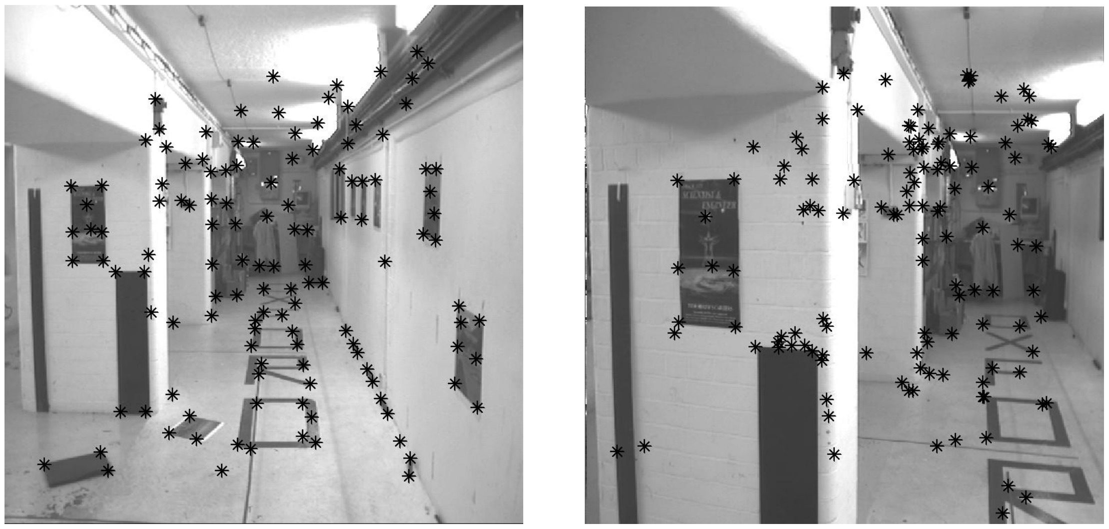


Figure 4.8: Fundamental matrix estimation for the *corridor* images. Frame 0 and frame 9 are shown along with all the 127 point matches (inliers and outliers).

***Corridor* Images**

To test our algorithm, we use two far apart frames, frames 0 and 9, from the well known *corridor* sequence. These images and the point matches are shown in Figure 4.8. The ground truth for this sequence is known and from this the inlier noise standard deviation is estimated as $\sigma_t = 0.88$. We compare the performance of pbM with RANSAC and MSAC. Both RANSAC and MSAC were tuned to the optimal value of $\sigma_{opt} = 1.96\sigma_t$. To simulate the real situation RANSAC and MSAC were also run after tuning them to the MAD scale estimate of (4.24).

Points were matched using the method of [47] and 500 elemental subsets were randomly generated. This gives 127 points with 58 inliers. Large portions of the first image are not visible in the second image and these points get mismatched. Like in the previous section, the performance of the estimators is compared based on the number of true inliers among points classified as inliers and the ratio of the standard deviation of the selected points to that of the true inlier noise. The results of the various estimators are shown in Table 4.3.

It is clear that pbM outperforms RANSAC and MSAC in spite of being user independent. The points retained as inliers by pbM are shown in Figure 4.9. True inliers are

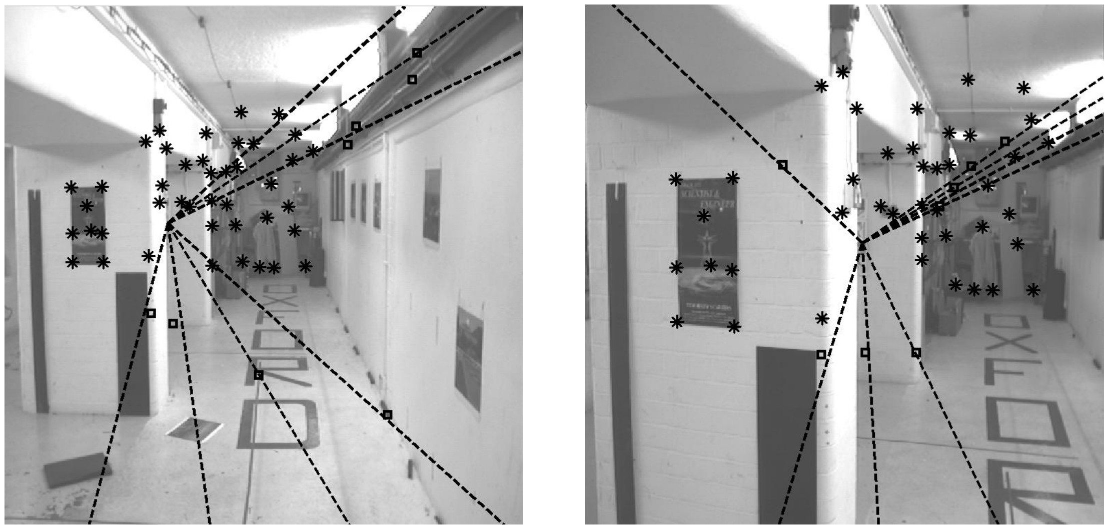


Figure 4.9: Results of fundamental matrix estimation for the *corridor* images. The 66 inliers returned by pbM and epipolar lines of the 8 outliers misclassified as inliers are shown. The reason for the misclassifications is explained in the text.

Table 4.3: Performance Comparison - *Corridor* Image Pair

	selected points/true inliers	σ_{in}/σ_t
RANSAC(σ_{opt})	35/30	12.61
MSAC(σ_{opt})	11/8	9.81
RANSAC(σ_{mad})	103/52	15.80
MSAC(σ_{mad})	41/18	9.76
pbM	66/58	1.99

shown as asterisks. Eight mismatches have been classified as inliers and these are shown as squares along with their epipolar lines. For these points, the epipolar lines pass very close to the mismatched points *or* one of the points lies close to the epipoles. In such cases the epipolar constraint (4.28) is satisfied. Since this is the only constraint that is being enforced, the system cannot detect such mismatches and these few mismatches are labeled inliers.

The fundamental matrix between two images should be of rank-2. This condition is usually ignored by robust regression algorithms. Once a satisfactory inlier/outlier dichotomy has been obtained, more complex estimation methods are applied to the inliers while enforcing the rank-2 constraint. Consequently, the fundamental matrix estimate returned by most robust regression algorithms are not good estimates of the

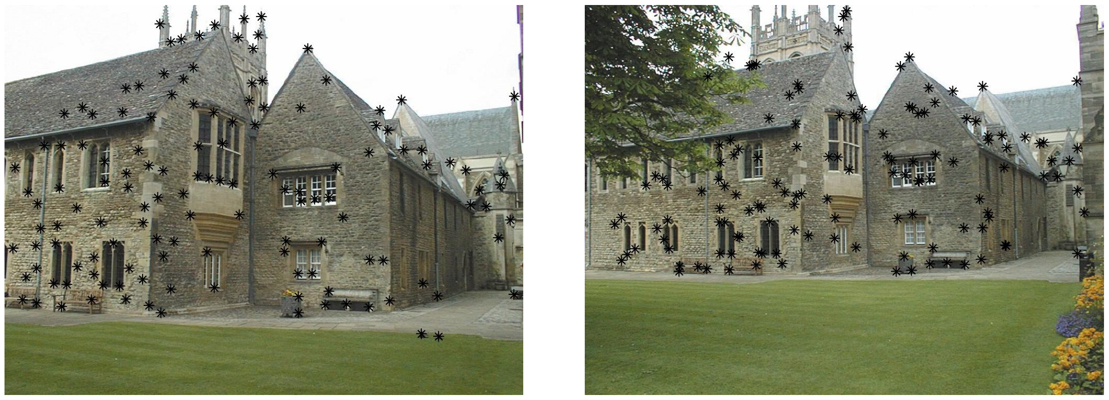


Figure 4.10: Fundamental matrix estimation for the *Merton College* images. Frames 0 and 2 are shown along with all the 135 point matches (inliers and outliers).

true fundamental matrix. In [14] a different version of pbM, which does not account for the heteroscedasticity of the data, was used for robust fundamental matrix estimation. In this case, it was found that pbM gives good inlier/outlier dichotomies but incorrect estimates of the fundamental matrix.

The heteroscedastic pbM algorithm discussed here nearly satisfies the rank-2 constraint even though it is not explicitly enforced. This is because the heteroscedastic nature of the noise is accounted for and our estimate is very close to the true fundamental matrix. For the estimate returned by pbM, the ratio of the second singular value to the third singular value is of the order of 10000. The singular values of the fundamental matrix estimated by pbM are 17.08, 2.92×10^{-2} and 5.61×10^{-7}. The epipoles computed from the estimated fundamental matrix also matched the ground truth epipoles.

***Merton College* Images**

We also tested pbM on two images from the *Merton college* data set from Oxford. The two images are shown in Figure 4.10. Points were matched using the method of [47] which gives 135 points with 68 inliers. For the robust estimation, 500 elemental subsets were used.

The fundamental matrix returned by pbM was close to the available ground truth estimate. It nearly satisfies the rank-2 constraint like in the previous example. The

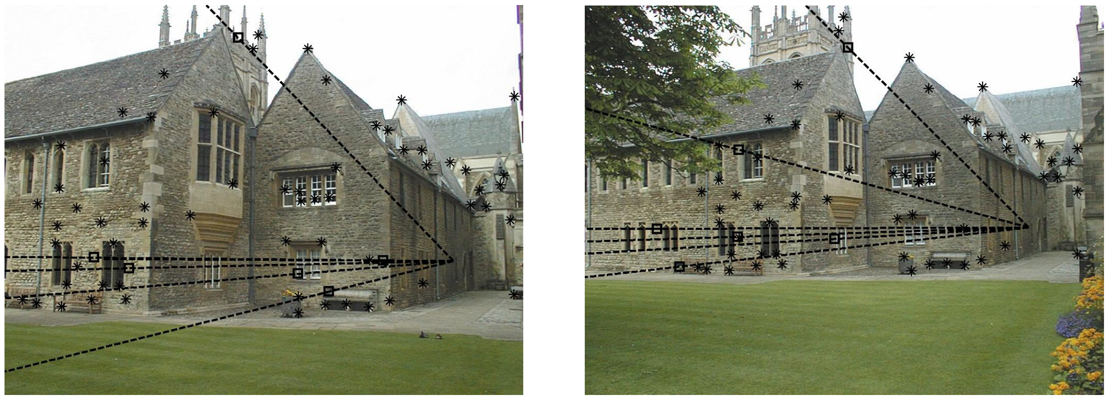


Figure 4.11: Results of fundamental matrix estimation for the *Merton College* images. The 68 inliers returned by pbM and epipolar lines of the 6 outliers misclassified as inliers are shown. The reason for the misclassifications is explained in the text.

singular values of the estimated fundamental matrix are 27.28, 1.83×10^{-2} and 7.38×10^{-8}. Six outliers are misclassified since they are mismatched to points lying on the correct epipolar line.

Table 4.4: Performance Comparison - *Merton College* Image Pair

	selected points/true inliers	σ_{in}/σ_t
RANSAC(σ_{opt})	21/27	10.962
MSAC(σ_{opt})	10/2	0.701
RANSAC(σ_{mad})	32/11	3.094
MSAC(σ_{mad})	43/32	13.121
pbM	68/62	0.92

Valbonne **Images**

The system was tested on another image pair taken from the *Valbonne* sequence. The images and the point matches are shown in Figure 4.12. The true matches were selected by manual inspection to obtain the ground truth. The standard deviation of the inlier noise was fond to be $\sigma_t = 0.072$. Of the 85 points matched, 42 were inliers and the rest are mismatches. The results of the various estimators are compared in Table 4.5. Again, pbM does better than RANSAC and MSAC in spite of being user-independent. The rank-2 constraint is satisfied with the ratio between the second and third singular values of the fundamental matrix being of the order of 10000. The epipoles from the

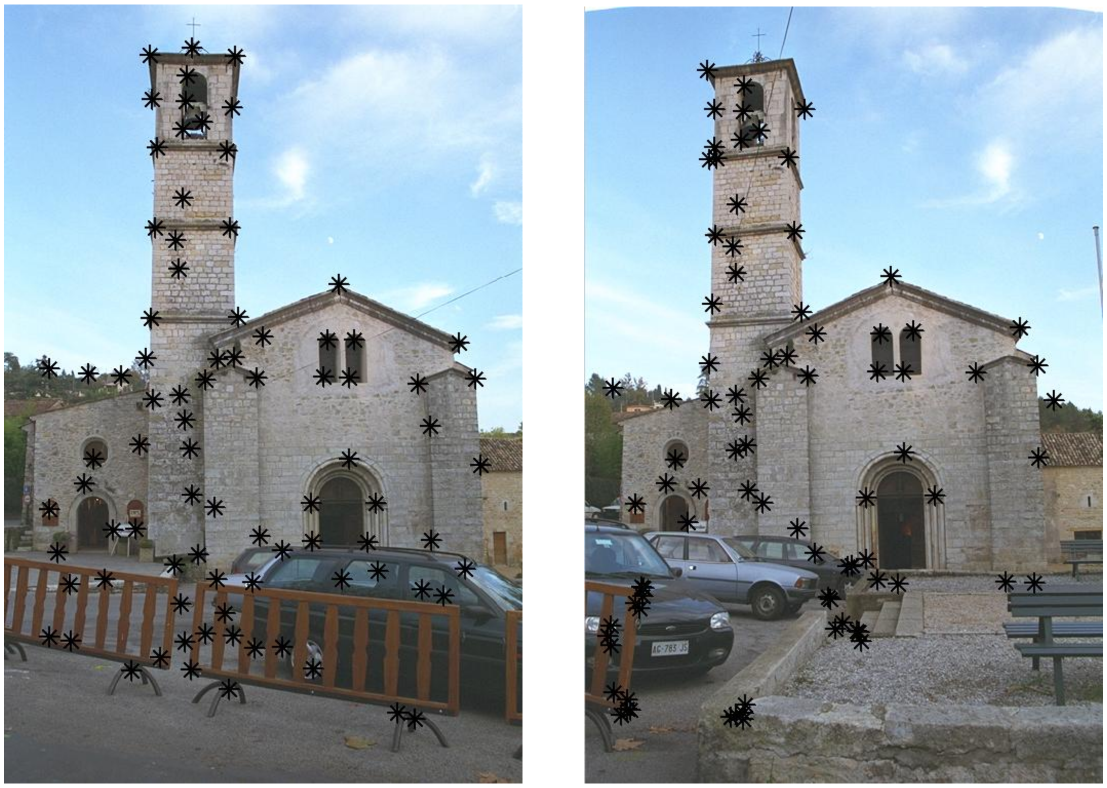


Figure 4.12: Fundamental matrix estimation for the *Valbonne* sequence. Both images are shown along with the 85 point matches (inliers and outliers).

estimated fundamental matrix also match the true epipoles.

Table 4.5: Performance Comparison - *Valbonne* Image Pair

	selected points/true inliers	σ_{in}/σ_t
RANSAC(σ_{opt})	31/26	7.36
MSAC(σ_{opt})	9/6	6.93
RANSAC(σ_{mad})	46/37	17.73
MSAC(σ_{mad})	27/21	11.82
pbM	45/42	0.97

In our examples, the epipoles lie within the image. As the epipoles move towards the line at infinity, the geometry becomes more degenerate and fundamental matrix estimation becomes more ill-posed. Under these conditions, pbM continues to give a good inlier-outlier dichotomy but the fundamental matrix estimate becomes more inaccurate. However, in practice, once a good dichotomy is obtained a more complex non-robust estimator such as HEIV [84] is employed.

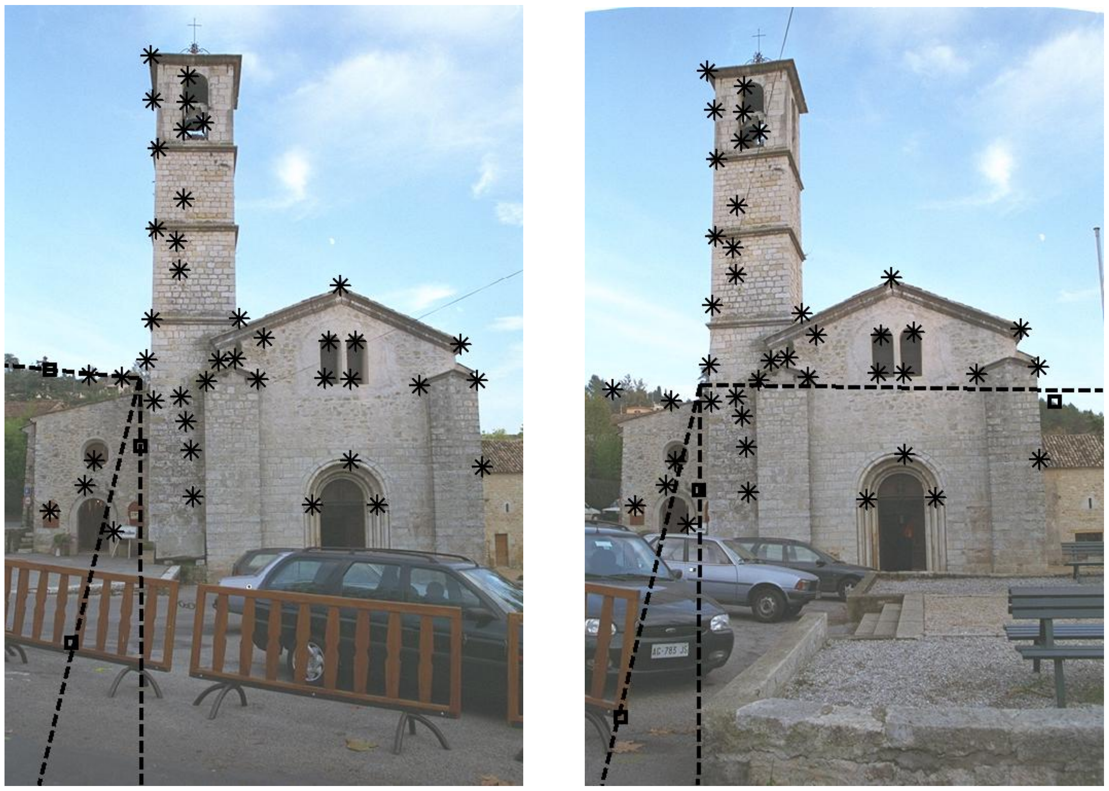


Figure 4.13: Results of fundamental matrix estimation for the *Valbonne* sequence. The 45 inliers returned by pbM and epipolar lines of the 3 outliers misclassified as inliers are shown.

4.5.4 Affine Factorization

In the previous examples we used pbM for the estimation of a single structure in the presence of unstructured outliers, that is outliers not belonging to an other geometrical structure such as another affine motion or another fundamental matrix. One advantage of pbM is that it works well in the presence of structured outliers. We show this with the example of multibody factorization. As we said earlier in Section 3.6.1, factorization is based on the fact that the positions of points tracked over F frames of an uncalibrated affine camera define a feature vector in $\mathbb{R}^{2F}$. When viewed with an affine camera points sharing the same motion define a four-dimensional subspace (three-dimensional subspace if the object is linear). Using pbM for factorization requires the subspace estimation version of the pbM algorithm.

In the presence of multiple bodies we get multiple subspaces and this is known as the multibody factorization problem. As mentioned in Section 4.1.3, most other multibody factorization methods make assumptions about the data which may not be true. In fact, in [113], pbM was compared to GPCA [134] and subspace separation [117] and it was shown that the presence of outliers and degenerate motion subspaces does lead to the breakdown of GPCA and subspace separation.

We use multiple constraint pbM to solve the affine factorization problem with multiple moving bodies and in the presence of outliers. Unstructured outliers which do not lie in any motion subspace occur in the data set due to mismatched point tracks. Publicly available affine factorization data sets, such as Kanatani's [117] or the Hopkins 155 [127], do not have large image motion between the frames. Therefore, the number of outliers is low and their algorithms are designed to handle possible degeneracies between the different motion subspaces. We try to address the problem of robustness in the presence of large numbers of outliers which do not belong to any motion. Our sequences have large displacements between frames leading to more outliers. They also consist of few frames leading to degeneracies. For example, with three motions over four frames it is impossible to have independent subspaces since only eight independent vectors can exist in the space, while at least nine linearly independent vectors are required for each motion subspace to have an independent basis.

For affine factorization, each elemental subset consists of four point tracks across the F frames. These tracks give a $2F \times 4$ matrix. The matrix is centered and a three-dimensional basis of the column space of the centered matrix is found by singular value decomposition. All n measurements are projected along this three-dimensional basis and the 3D intercept is found as the maxima of the kernel density.

For pbM we used 1000 elemental subsets for estimating the first subspace, and 500 elemental subsets for estimating each further subspace. The ground truth was obtained by manually segmenting the feature points.

Lab **Sequence**

We used the point matching algorithm of [47] to track points across the frames. Two of the motions are degenerate and share one of the basis vectors. The data consists of 231 point tracked across five frames. Points in the background are detected as having

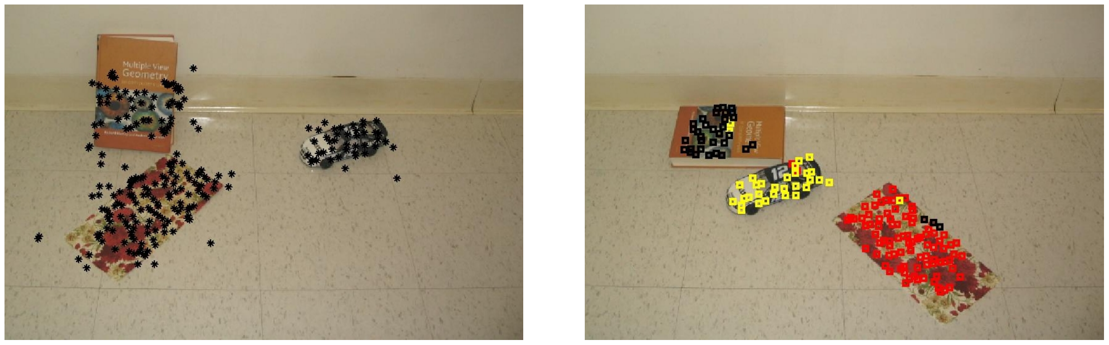


Figure 4.14: *Lab* sequence used for factorization. The three objects move independently and define different motion subspaces. The left image shows the first frame with all the points (inliers and outliers). The right shows the fifth frame with the points assigned to the three motions marked differently. The first motion $\mathbf{M_1}$ corresponds to the paper napkin, the second motion $\mathbf{M_2}$ to the car and $\mathbf{M}_3$ to the book.

no motion and removed. All 231 corners are plotted in the first frame in the image on the left in Figure 4.14. The three bodies move independently and have 83, 29 and 32 inliers and 87 outliers. Note, that the number of outliers is more than any group of inliers. The segmented results are shown in the right image in Figure 4.14 where the points assigned to the different motions are plotted differently on the fifth frame. The results of the segmentation are tabulated in Table 4.6. The first line says that of the 87 points returned by the system as inliers for the first motion (paper napkin), 83 are true inliers, two are on the second object (the car) and two are outliers.

Table 4.6: Segmentation Results of Factorization

	$\mathbf{M}_1$	$\mathbf{M}_2$	$\mathbf{M}_3$	*Out*
$\mathbf{M}_1$	83	2	0	2
$\mathbf{M}_2$	0	28	2	2
$\mathbf{M}_3$	0	0	31	4
Out	0	2	1	74

We used the same elemental subsets generated for pbM to get robust estimates using RANSAC and MSAC. Both RANSAC and MSAC pick a wrong subspace basis and do not return a good motion even for the first motion.

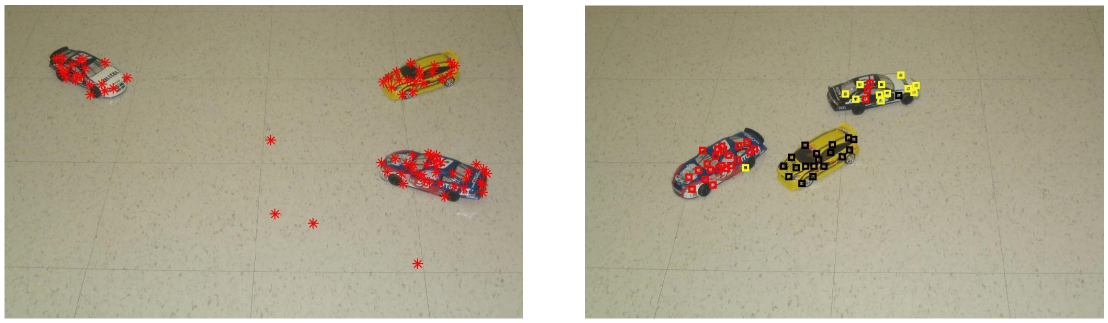


Figure 4.15: *Toy car* sequence used for factorization. The three cars move independently and define different motion subspaces, but the yellow and black cars define subspaces very close to each other. The left image shows the first frame with all the points (inliers and outliers). The right image shows the fourth frame with the points assigned to the three cars marked differently. The first motion $\mathbf{M_1}$ corresponds to the blue car, $\mathbf{M_2}$ to the yellow car and $\mathbf{M_3}$ to the black car.

***Toy Car* Sequence**

We use the point matching algorithm of [47] to track points across the frames. This data consists of 77 point tracked across four frames. As fewer frames are involved, the degeneracies between the subspaces are more pronounced. The first and the third objects define very similar motion subspaces. This makes it harder for algorithms like RANSAC where the scale is held constant. However, pbM adapts the scale to the current parameter estimate and manages to correctly segment the three motions. All 77 corners are plotted in the first frame in the image on the left in Figure 4.15. The three cars have 23, 18 and 17 inliers with 19 outliers. The segmented results are shown in the right image in Figure 4.15 where the points assigned to the different motion are plotted differently on the fourth frame. The results of the segmentation are tabulated in Table 4.7.

Table 4.7: Segmentation Results of Factorization

	$\mathbf{M}_1$	$\mathbf{M}_2$	$\mathbf{M}_3$	*Out*
$\mathbf{M}_1$	22	0	3	0
$\mathbf{M}_2$	0	18	1	0
$\mathbf{M}_3$	1	0	11	0
Out	0	0	2	19

4.6 Connection to Nonlinear Mean Shift

We have used both the nonlinear mean shift algorithm and pbM for affine motion estimation and multibody factorization. Although the two of them try to solve the same problems in different ways there are some significant differences between them.

Most importantly, nonlinear mean shift is a procedure for finding the local maxima of a kernel density over a manifold. Motion segmentation or robust estimation are applications of this algorithm but it can also be used for image filtering *etc*. On the other hand, pbM is a robust regression algorithm which makes the robust estimation procedure user-independent.

Even when viewed as a robust estimator, nonlinear mean shift differs from pbM in a few significant ways.

- Mean shift tries to find all the structures present in a single step without any specification of the number of different models present while pbM only finds the most dominant structure. Further structures are found by iteratively removing inliers and running pbM again. Therefore, pbM needs some knowledge of the number of structures present.

- For pbM, it is sufficient if a single hypothesis is generated from inliers since this will give a hypothesis close to the true model. Mean shift requires the presence of a mode at the true model parameters. Therefore, it is not sufficient for a single hypothesis to be close to the true parameters but a sufficient number of hypotheses need to be around the true parameters to make the mode significant enough to be detected. The validation step is a partial solution to the problem but it is not sufficient in the presence of large numbers of outliers.

- Mean shift requires the user to specify the bandwidth. In some cases performance is sensitive to the value of the bandwidth. The bandwidth can be thought of as a reflection of the scale of the noise, which pbM detects automatically. Therefore, mean shift does not offer the advantage of user independence which pbM does.

Some ideas on combining these two methods are discussed in Chapter 6.

Chapter 5

Fusion of Multiple Sources in Structure from Motion

Fusing information from multiple sources is a problem which has received a lot of attention in various fields of engineering. Given noisy estimates of measurements of a parameter from different sensors, fusion attempts to combine these estimates to find the best estimate of the parameter. Here we consider a slightly different problem where some of the sensors only supply partial information about the parameter being estimated. Another important aspect that is often overlooked is the fact that the constraint equation relating the given measurements and the parameters being estimated are nonlinear leading to heteroscedastic data measurements [84].

An application where it is necessary to account for these aspects of sensor fusion is the Structure-from-Motion (SFM) problem in computer vision. Estimating the three-dimensional structure of a scene from a series of two-dimensional images is the central problem of computer vision. Methods that try to solve this problem can be broadly classified into *causal* methods which only use information available from the past and *noncausal* schemes which attempt to infer the three-dimensional structure after all the data has been collected. If the camera has not been calibrated and the internal parameters of the camera are unknown, the reconstruction of the algorithm differs from the true reconstruction by some projective transformation. If the calibration of the camera is known, the reconstruction can be improved so that it differs from the true structure by a rigid body transformation and scaling.

Due to the nature in which the data is collected and used in noncausal schemes, they cannot be used for applications in which pose estimates are needed in real time. An example of such an application is augmented reality (AR). Augmented reality systems enhance a user's interaction with the real world through additional information generated by computer models of the world. In AR systems these scene enhancements are achieved through graphics generated from geometric models of real and virtual objects. In order for the alignment between the real and virtual objects, it is necessary to know the pose and the internal parameters of the camera. This is known as the 3D tracking problem. In practice, the camera is calibrated offline and its internal parameters are known. The availability of fast processors and frame grabbers, have made vision based trackers extremely popular and there are similarities between the causal, calibrated, structure-from-motion (SFM) problem in computer vision and 3D tracking for AR.

In this paper we propose a statistical method to combine information from multiple sources for scene reconstruction. We achieve this fusion by modifying the heteroscedastic errors-in-variables regression algorithm [84]. For our experiments, we consider a system consisting of a single camera and a rigidly attached gyroscope. The gyroscope gives noisy estimates of the rotation of the camera and this is combined with the image information from the camera.

In Section 5.1 we review some of the previous work on this problem. There is a large amount of work devoted to solving SFM and we cannot do justice to all of it. We only discuss some of the more significant work in this direction. In Section 5.2 we introduce the heteroscedastic errors-in-variables (HEIV) algorithm for bilinear regression and in Section 5.3 we discuss a camera tracking system based on HEIV. This system infers scene structure based on image information. The HEIV formulation is modified in Section 5.4 and a new algorithm is derived which accounts for multiple sources of information. The results of our algorithm are presented in Section 5.5.

5.1 Previous Work

Structure from motion (SFM) is a central problem in computer vision which has been extensively analyzed. The goal of SFM is to estimate the 3D shape and motion of a set of features given the velocity of their perspective projections in the image (optical flow) or the correspondences between different projections (point correspondences).

There exists a large amount of literature which tries to reconstruct the scene given eight or more point correspondences between two images. These algorithms proceed by estimating the epipolar geometry or trifocal tensor between frames, and then transforming all the reconstructions to a common reference frame [95, 101]. If the camera is calibrated, then the essential matrix is estimated rather than the fundamental matrix. These methods are based on the geometrical relations between 3D feature points and their projections into the frames. The geometry of this has been extensive studied [57] and is very well understood. In practice, the geometrical relations do not hold exactly. The estimates of the projections are noisy and do not satisfy the required constraints exactly and computationally expensive nonlinear optimization methods, such as bundle adjustment [126], are required to find the *optimal* estimates of scene structure. However, global optimization is done at the last step to get good accuracy, and these methods are not causal. A number of statistically optimal methods for geometric estimation have also been developed specifically to account for the nonlinear geometric relations [65, 82]. These methods can be used in a causal framework [115], but fail to account for the dynamics of the camera motion.

The causal, calibrated, structure from motion problem is known as the camera tracking problem in the augmented reality community. The aim of camera tracking is to estimate the pose of the camera so that the virtual objects can be aligned with the real objects. While estimating structure is one of the aims of SFM algorithms, in camera tracking, structure estimation is considered just a step to get the camera pose. Consequently, a number of algorithms have been developed which give the camera tracking extra information, such as 3D models of some of the scene, the world positions of easily identifiable markers in the scene etc. [46, 29, 76, 103, 115]. Simon et al [103] assume the presence of planar structures in the scene to perform camera tracking. However, the assumption of planarity is a special case and is not true always. Vacchetti et al [76] combine offline information from keyframes with online information deduced from a traditional frame-to-frame tracking approach. For structure recovery, they use a two frame approach where point correspondences across two frames are found and triangulated in 3D. To reduce the number of unknowns in this procedure, they use a 3D model of the scene and the knowledge that all image points must lie on the surface of the 3D model. Genc et al [46] use a learning based method where the initial coordinate system is defined by markers but the system is allowed to move away from the markers. The scene structure is sequentially estimated frame-to-frame using previously triangulated scene features. This method is similar to the *Simultaneous Localization and Mapping* (SLAM) algorithm of [29]. In [115] it was shown that the results of [46] can be improved by accounting for nonlinearities present in the system by using the Heteroscedastic-Errors-In-Variables (HEIV) estimator [82, 84] to solve the nonlinear regression for camera pose and structure recovery. Using the balanced HEIV estimator instead of nonlinear optimization techniques gives better results, as we show in Section 5.3.

Recently, there has also been some interest in using multiple sensors to improve the accuracy of camera tracking systems. Some methods use multiple cameras [61], inertial sensors [61] or optical trackers [60] to improve the scene structure inferred by a single camera. When using multiple sources it is necessary to combine the various data sources. Each of these data sources is inherently noisy, and they measure different parameters of the motion. For example, gyroscopes measure the rotation but not the translation.

As we discuss in Section 5.4.1, these methods are similar to a class of causal SFM methods which are usually referred to as *recursive* or *filtering based* methods. Recursive methods exist for the estimation of motion given the scene structure [10], estimation of structure given motion [85] or the estimation of both simultaneously [5, 18]. These methods approach the problem from a filtering perspective and use Kalman filters to model the dynamics of the system. The theoretical properties of the filtering approach have also been well studied and it is shown that a minimal parametrization of the problem is possible, which is stable and observable [18]. The filtering based approaches account for the dynamics of the camera motion. However, these methods cannot be expected to give good results due to the nonlinear nature of the projection equations.

The method we propose here is an extension of the system of [115]. The HEIV formulation is modified to allow the system to fuse image information with camera motion information obtained from inertial sensors. Doing this optimally allows us to account for the dynamics of the motion. We do not account for outliers in our system since they are too few to cause serious errors in the pose estimation and reconstruction. This assumption is valid for the camera motions we consider in this paper. For fast moving cameras, where a wide baseline exists between consecutive frames, outliers in the image tracking can become a problem. In this case robust methods can easily be included in the system proposed here, like in system of [115].

5.2 Heteroscedastic Errors-in-Variables Algorithm

Most computer vision problems require the estimation of a set of parameters from a set of noisy measurements. This is done by enforcing a *constraint* equation which relates the measurements to a set of parameters while assuming a *noise model* which characterizes the errors affecting the given measurements. The constraint equation and noise model are collectively known as a *statistical model.* The constraint enforces an implicit relationship between the true values of the measurements, $\mathbf{z}_{io}$ and the true parameter $\boldsymbol{\theta}_o$. We further assume, that the constraint can be factored into two parts. The first part is a nonlinear function of the measurement vectors and the second is the parameter vector. Such constraints are known as *separable* constraints. For most vision problems, the constraints are separable and the nonlinear function is a polynomial.

$$\boldsymbol{f}(\boldsymbol{z}_{io}, \boldsymbol{\theta}_o) = \Phi(\boldsymbol{z}_{io})\boldsymbol{\theta}_o = \mathbf{0} \qquad \Phi(\cdot) \in \mathbb{R}^{m \times p} \qquad \boldsymbol{\theta}_o \in \mathbb{R}^p \tag{5.1}$$

The functions $\Phi(\boldsymbol{z}_{io})$ are known as *carrier* vectors. The constraint (5.1) has a multiplicative ambiguity with respect to a nonzero constant. This is removed by assuming that $\boldsymbol{\theta}_0$ has unit norm. In some problems, it might be necessary to further constrain that parameter vector. We discuss this problem later in Section 5.2.3.

The noise corrupting the measurements is taken to be additive. In the most general case, the characteristics of the noise depend on the data point, and such noise is said to be heteroscedastic

$$\boldsymbol{z}_i = \boldsymbol{z}_{io} + \delta\boldsymbol{z}_i \qquad \delta\boldsymbol{z}_i \sim GI(\mathbf{0}, \sigma^2\boldsymbol{C}_{\boldsymbol{z}_i}) \qquad i = 1, \ldots, n \qquad \boldsymbol{z}_{io} \in \mathbb{R}^s \tag{5.2}$$

where GI stands for a general, symmetric and independent distribution whose first two central moments are available. The subscript 'o' is used to distinguish the ideal value of a quantity from its noisy measurement. The covariance matrices are known up to a common global scaling of σ^2.

Note the similarity to the problem definitions of (4.1) and (4.16). The main difference is that here we assume all the points are inliers. We also explicitly model the relation between the low dimensional data vectors $\mathbf{z}_i$ and the higher dimensional carrier vectors $\Phi(\boldsymbol{z}_i)$ which appear in the regression.

5.2.1 An Approach Through Linearization

The algorithm discussed here is described in [82]. A more complete version of the algorithm, its derivation and its applications can be found in [84]. Here we give a very brief version of the derivation of the algorithm.

The unobservable noise-free data $\boldsymbol{z}_{io}, i = 1, \ldots, n$ and the unknown true parameter vector $\boldsymbol{\theta}_o$ satisfy the constraint (5.1). From the measurements $\boldsymbol{z}_i$ which have been corrupted by additive heteroscedastic noise find the estimates $\hat{\boldsymbol{z}}_i, i = 1, \ldots, n$ and $\hat{\boldsymbol{\theta}}$ such that

$$\boldsymbol{f}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{\theta}}) = \Phi(\hat{\boldsymbol{z}}_i)\hat{\boldsymbol{\theta}} = (\mathbf{e_p} \otimes \hat{\boldsymbol{\theta}})\varphi(\hat{\boldsymbol{z}}_i) = \mathbf{0} \qquad \|\hat{\boldsymbol{\theta}}\| = 1 \tag{5.3}$$

where, $\varphi(\boldsymbol{z}_i) = \mathbf{vec}(\Phi(\boldsymbol{z}_i)^T)$ and $\otimes$ is the Kronecker product of two matrices [52]. The estimates are obtained by minimizing the objective function

$$\mathcal{J}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{z}}_1, \ldots, \hat{\boldsymbol{z}}_n) = \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i)^T \boldsymbol{C}^{+}_{\boldsymbol{z}_i}(\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i) \tag{5.4}$$

which is the sum of squared Mahalanobis distances between a measurement and its corrected value. To account for possible degeneracies in the covariance matrices, the pseudo inverse rather than the inverse of the matrices is used and this is denoted by the superscript '+'. The global scaling σ^2 has been dropped since it does not influence the minimum of the objective function. If the measurement noise is known to be normally distributed then this estimation process is equivalent to maximum-likelihood estimation.

To account for the constraints, we introduce the Lagrange multipliers $\boldsymbol{\eta}_i \in \mathbb{R}^m$ and minimize

$$\mathcal{J}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{z}}_1, \ldots, \hat{\boldsymbol{z}}_n) = \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i)^T \boldsymbol{C}^{+}_{\boldsymbol{z}_i}(\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i) + \sum_{i=1}^{n}\boldsymbol{\eta}_i^T \boldsymbol{f}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{\theta}}). \tag{5.5}$$

At the minimum of the above cost function, the parameter estimates must obey

$$\boldsymbol{J}_{\mathcal{J}|\boldsymbol{z}_i}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{z}}_1, \ldots, \hat{\boldsymbol{z}}_n) = \mathbf{0} \qquad i = 1, \ldots, n \tag{5.6}$$

$$\boldsymbol{J}_{\mathcal{J}|\boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{z}}_1, \ldots, \hat{\boldsymbol{z}}_n) = \mathbf{0} \tag{5.7}$$

where, $\boldsymbol{J}_{\mathcal{J}|\mathbf{u}}$ is the Jacobian of the scalar objective function with respect to the vector $\mathbf{u}$. Evaluating (5.6) and (5.7) and after a non-trivial amount of algebraic manipulation, we get [84]

$$\boldsymbol{J}_{\mathcal{J}|\boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}) = \left[\boldsymbol{S}(\hat{\boldsymbol{\theta}}) - \boldsymbol{C}(\hat{\boldsymbol{\theta}})\right]\hat{\boldsymbol{\theta}} = \mathbf{0} \tag{5.8}$$

where, the *weighted scatter matrix* $S(\hat{\boldsymbol{\theta}})$ is

$$\boldsymbol{S}(\hat{\boldsymbol{\theta}}) = \sum_{i=1}^{n}\Phi(\boldsymbol{z}_i)^T \boldsymbol{C}_f(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}})^{+}\Phi(\boldsymbol{z}_i) \tag{5.9}$$

the *weighted covariance matrix* is

$$\boldsymbol{C}(\hat{\boldsymbol{\theta}}) = \sum_{i=1}^{n}(\boldsymbol{\eta}_i \otimes \mathbf{e_p})^T \boldsymbol{C}_{\varphi}(\boldsymbol{z}_i)(\boldsymbol{\eta}_i \otimes \mathbf{e_p}). \tag{5.10}$$

The matrix $\boldsymbol{C}_f(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}})$ is the first order approximation of the covariance of the expression $\boldsymbol{f}(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}})$ computed by error propagation from $\boldsymbol{z}_i$ to $\boldsymbol{f}(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}})$. Given $\boldsymbol{z}_i$ and its covariance $\boldsymbol{C}_{\boldsymbol{z}_i}$, we have

$$\boldsymbol{C}_{\boldsymbol{f}}(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}}) = \boldsymbol{J}^T_{\boldsymbol{f}|\boldsymbol{z}_i}(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}})\boldsymbol{C}_{\boldsymbol{z}_i}\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}}) \ . \tag{5.11}$$

Similarly, the matrix $\boldsymbol{C}_{\varphi}(\boldsymbol{z}_i)$ is the first order approximation of the covariance of the expression $\varphi(\boldsymbol{z}_i)$ computed by error propagation from $\boldsymbol{z}_i$ to $\varphi(\boldsymbol{z}_i)$

$$\boldsymbol{C}_{\varphi}(\boldsymbol{z}_i) = \boldsymbol{J}^T_{\varphi|\boldsymbol{z}_i}(\boldsymbol{z}_i)\boldsymbol{C}_{\boldsymbol{z}_i}\boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\boldsymbol{z}_i) \ . \tag{5.12}$$

The Lagrange multipliers are given by

$$\boldsymbol{\eta}_i = \boldsymbol{C}_{\boldsymbol{f}}(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}})^{+}\boldsymbol{f}(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}}) \ . \tag{5.13}$$

The problem (5.8) can be rewritten as a generalized eigenproblem

$$\boldsymbol{S}(\hat{\boldsymbol{\theta}})\hat{\boldsymbol{\theta}} = \boldsymbol{C}(\hat{\boldsymbol{\theta}})\hat{\boldsymbol{\theta}} \tag{5.14}$$

and this is called the *heteroscedastic errors-in-variables* (HEIV) equation.

5.2.2 The Heteroscedastic Errors-in-Variables Algorithm

It is difficult to solve the HEIV equation due to the dependence of $\boldsymbol{S}(\hat{\boldsymbol{\theta}})$ and $\boldsymbol{C}(\hat{\boldsymbol{\theta}})$ on $\hat{\boldsymbol{\theta}}$. The algorithm proceeds by iteratively improving the estimate of $\boldsymbol{\theta}_o$.

Let $\check{\boldsymbol{z}}_i$ and $\hat{\boldsymbol{\theta}}^{[k]}$ be estimates of $\boldsymbol{z}_{io}$ and $\boldsymbol{\theta}_o$ after k iterations. Initially, in the absence of any prior information, we assume $\check{\boldsymbol{z}}_i = \boldsymbol{z}_i$ and $\hat{\boldsymbol{\theta}}$ is chosen randomly. For the $k+1$-th iteration the expressions for the scatter matrix and covariance matrix involve linearization around $\check{\boldsymbol{z}}_i$. These matrices are given by

$$\boldsymbol{S}(\hat{\boldsymbol{\theta}}^{[k]}) = \sum_{i=1}^{n} \Phi(\boldsymbol{z}_i)^T \boldsymbol{C}_f(\check{\boldsymbol{z}}_i, \hat{\boldsymbol{\theta}}^{[k]})^+ \Phi(\boldsymbol{z}_i) \tag{5.15}$$

$$\boldsymbol{C}(\hat{\boldsymbol{\theta}}^{[k]}) = \sum_{i=1}^{n} (\boldsymbol{\eta}_i \otimes \mathbf{e_p})^T \boldsymbol{C}_\varphi(\check{\boldsymbol{z}}_i)(\boldsymbol{\eta}_i \otimes \mathbf{e_p}) \tag{5.16}$$

where the Lagrange multipliers are

$$\boldsymbol{\eta}_i = \boldsymbol{C}_{\boldsymbol{f}}(\check{\boldsymbol{z}}_i, \hat{\boldsymbol{\theta}}^{[k]})^+ \boldsymbol{f}(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}}^{[k]}) \ . \tag{5.17}$$

Note, that the values of $\check{\boldsymbol{z}}_i$ are updated for each iteration, while $\boldsymbol{z}_i$ is constant. The matrices defined above depend on both $\boldsymbol{z}_i$ and $\check{\boldsymbol{z}}_i$. This comes out from the derivation [84]. This generalized eigenproblem [49, p.394]

$$\boldsymbol{S}(\hat{\boldsymbol{\theta}}^{[k]})\hat{\boldsymbol{\theta}}^{[k+1]} = \lambda \boldsymbol{C}(\hat{\boldsymbol{\theta}}^{[k]})\hat{\boldsymbol{\theta}}^{[k+1]} \tag{5.18}$$

is solved by converting it into a generalized singular value decomposition [49, p.471] in terms of the matrix square roots of $\boldsymbol{S}(\hat{\boldsymbol{\theta}})$ and $\boldsymbol{C}(\hat{\boldsymbol{\theta}})$. The generalized eigenvector corresponding to the smallest generalized eigenvalue λ is taken to be $\hat{\boldsymbol{\theta}}^{[k+1]}$. The iteration can also be solved by a standard eigenproblem, but the generalized eigenproblem is more numerically stable gives better performance in practice. The estimates $\check{\boldsymbol{z}}_i$ are improved according to

$$\hat{\boldsymbol{z}}_i = \boldsymbol{z}_i - \boldsymbol{C}_{\boldsymbol{z}_i} \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \hat{\boldsymbol{\theta}}^{[k+1]}) \boldsymbol{C}_{\boldsymbol{f}}(\check{\boldsymbol{z}}_i, \hat{\boldsymbol{\theta}}^{[k+1]})^+ \boldsymbol{f}(\boldsymbol{z}_i, \hat{\boldsymbol{\theta}}^{[k+1]}). \tag{5.19}$$

The values $\hat{\boldsymbol{z}}_i$ are used as the estimates $\check{\boldsymbol{z}}_i$ for the next iteration. The scatter matrix and covariance matrix are computed with the estimates $\hat{\boldsymbol{z}}_i$ and process is repeated till convergence.

Convergence occurs when the smallest singular value λ is close enough to one. The values of λ converge to one quickly from below. In practice, the method usually requires 4-5 iterations.

The HEIV equation is the most general expression for solving separable EIV problems. Special cases of the HEIV equation appear in numerous techniques commonly used in computer vision. In [84] it is shown that methods such as generalized total least squares [131], the Sampson method [100, 57] and renormalization [65] are special cases of the HEIV formulation.

5.2.3 Enforcing Ancillary Constraints

The parameter estimate $\hat{\boldsymbol{\theta}}$ may be required to satisfy additional constraints. The method for enforcing these constraints is discussed below. A more complete discussion can be found in [84]. We present an example of such constraints in Section 5.2.4. Let the constraints be denoted as

$$\boldsymbol{\zeta}(\hat{\boldsymbol{\theta}}) = \mathbf{0}, \quad \boldsymbol{\zeta}(\cdot) \in \mathbb{R}^t \; . \tag{5.20}$$

To obtain constrained estimates, HEIV proceeds by first doing an unconstrained estimation of $\hat{\boldsymbol{\theta}}$. At convergence, the *a posteriori* covariance of $\hat{\boldsymbol{\theta}}$ is given by [84]

$$\boldsymbol{C}_{\hat{\boldsymbol{\theta}}} = \sigma_\nu^2 \left[\boldsymbol{S}(\hat{\boldsymbol{\theta}}) - \boldsymbol{C}(\hat{\boldsymbol{\theta}})\right]^+ \tag{5.21}$$

An unbiased estimate of the scale of the noise σ_ν^2 can be obtained from the residual error [84]. In the following procedure, it is sufficient to know $\boldsymbol{C}_{\hat{\boldsymbol{\theta}}}$ up to scale, and therefore we use $\boldsymbol{C}_{\hat{\boldsymbol{\theta}}} = \left[\boldsymbol{S}(\hat{\boldsymbol{\theta}}) - \boldsymbol{C}(\hat{\boldsymbol{\theta}})\right]^+$. Given a parameter estimate $\hat{\boldsymbol{\theta}}$, which does not satisfy the constraints, and its *a posteriori* covariance, the additional constraints are enforced by finding a $\hat{\hat{\boldsymbol{\theta}}}$ such that

$$\hat{\hat{\boldsymbol{\theta}}} = \arg\min_{\boldsymbol{\theta}} (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T \boldsymbol{C}_{\hat{\boldsymbol{\theta}}}^+ (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}) \tag{5.22}$$

subject to $\boldsymbol{\zeta}(\hat{\boldsymbol{\theta}}) = \mathbf{0}$. Introducing the Lagrange multipliers $\boldsymbol{\eta} \in \mathbb{R}^t$ we get the cost function

$$\mathcal{J} = \frac{1}{2}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T \boldsymbol{C}^{+}_{\hat{\boldsymbol{\theta}}}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}) + \boldsymbol{\eta}^T \boldsymbol{\zeta}(\hat{\boldsymbol{\theta}}) \ . \tag{5.23}$$

The solution $\hat{\hat{\boldsymbol{\theta}}}$ is found by setting the Jacobian $\boldsymbol{J}_{\mathcal{J}|\boldsymbol{\theta}}(\hat{\hat{\boldsymbol{\theta}}}) = \mathbf{0}$. Using a first order Taylor expansion of $\boldsymbol{\zeta}(\cdot)$ and some simple algebra we get

$$\hat{\hat{\boldsymbol{\theta}}} = \hat{\boldsymbol{\theta}} - \boldsymbol{C}_{\hat{\boldsymbol{\theta}}} \boldsymbol{J}_{\boldsymbol{\zeta}|\boldsymbol{\theta}}(\hat{\hat{\boldsymbol{\theta}}}) \left[\boldsymbol{J}_{\boldsymbol{\zeta}|\boldsymbol{\theta}}(\hat{\hat{\boldsymbol{\theta}}})^T \boldsymbol{C}_{\hat{\boldsymbol{\theta}}} \boldsymbol{J}_{\boldsymbol{\zeta}|\boldsymbol{\theta}}(\hat{\hat{\boldsymbol{\theta}}}) \right]^{+} \boldsymbol{\zeta}(\hat{\boldsymbol{\theta}}) \tag{5.24}$$

The above equation does not have a closed for solution. Therefore, the solution is obtained by doing the following iteration

$$\hat{\hat{\boldsymbol{\theta}}}^{[j+1]} = \hat{\hat{\boldsymbol{\theta}}}^{[j]} - \boldsymbol{C}_{\hat{\boldsymbol{\theta}}} \boldsymbol{J}_{\boldsymbol{\zeta}|\boldsymbol{\theta}}(\hat{\hat{\boldsymbol{\theta}}}^{[j]}) \left[\boldsymbol{J}_{\boldsymbol{\zeta}|\boldsymbol{\theta}}(\hat{\hat{\boldsymbol{\theta}}}^{[j]})^T \boldsymbol{C}_{\hat{\boldsymbol{\theta}}} \boldsymbol{J}_{\boldsymbol{\zeta}|\boldsymbol{\theta}}(\hat{\hat{\boldsymbol{\theta}}}^{[j]}) \right]^{+} \boldsymbol{\zeta}(\hat{\boldsymbol{\theta}}^{[j]}) \ . \tag{5.25}$$

The complete derivation can be found in [84].

We now discuss two applications in computer vision and their formulations as HEIV problems. These problems occur as part of the camera tracking system for augmented reality [115].

5.2.4 Camera Pose Estimation

In camera pose estimation we are given a set of 3D point coordinates, $\boldsymbol{X}_i \in \mathbb{R}^3, i = 1, \ldots, n$, and a corresponding set of 2D image coordinates $\boldsymbol{x}_i = [x_{i1} \;\; x_{i2}]^T \in \mathbb{R}^2, i = 1, \ldots, n$. We would like to estimate the camera pose $\boldsymbol{P} = [\boldsymbol{R} \;\; \boldsymbol{t}]$ such that $\boldsymbol{R} \in \mathbb{R}^{3\times 3}$ is orthogonal and

$$\begin{bmatrix} \boldsymbol{x}_i \\ 1 \end{bmatrix} \propto \boldsymbol{P} \begin{bmatrix} \boldsymbol{X}_i \\ 1 \end{bmatrix} \tag{5.26}$$

where, $\propto$ implies projective equivalence. In practice, this constraint does not hold exactly due to the additive noise corrupting the estimates. For ideal data the constraint can be written as

$$\begin{bmatrix} \boldsymbol{X}_{io}^T & \mathbf{0}^T & -x_{i1o}\boldsymbol{X}_{io}^T \\ \mathbf{0}^T & \boldsymbol{X}_{io}^T & -x_{i2o}\boldsymbol{X}_{io}^T \end{bmatrix} \boldsymbol{p} = \Phi(\boldsymbol{x}_{io}, \boldsymbol{X}_{io})\boldsymbol{p} = \mathbf{0} \tag{5.27}$$

where, $\boldsymbol{p} = \mathbf{vec}(\boldsymbol{P}^T)$ is the 12-dimensional vector obtained by stacking rows of the matrix $\boldsymbol{P}$.

The data vector for HEIV is the five dimensional vector $\boldsymbol{z}_i \in \mathbb{R}^5,\ i = 1, \ldots, n$ representing 3D-2D point matches. Therefore,

$$\boldsymbol{z}_i = \begin{bmatrix} \boldsymbol{X}_i \\ \boldsymbol{x}_i \end{bmatrix} . \tag{5.28}$$

The vector $\boldsymbol{z}_i$ is assumed to be a noise corrupted version of the true vector $\boldsymbol{z}_{io}$ and the covariance of this additive noise is $\sigma^2 \boldsymbol{C}_i \in \mathbb{R}^{5\times5},\ i = 1, \ldots, n$, where σ^2 is the unknown scaling of the covariance matrices. The nonlinear map $\Phi : \mathbb{R}^5 \rightarrow \mathbb{R}^{2\times12}$ is given by (5.27).

To enforce the orthogonality, we use the following set of constraints on the elements of $\boldsymbol{p}$,

$$\begin{array}{ll} \boldsymbol{p}_1^T\boldsymbol{p}_2 = 0 & \boldsymbol{p}_1^T\boldsymbol{p}_1 = \boldsymbol{p}_2^T\boldsymbol{p}_2 \\ \boldsymbol{p}_2^T\boldsymbol{p}_3 = 0 & \boldsymbol{p}_2^T\boldsymbol{p}_2 = \boldsymbol{p}_3^T\boldsymbol{p}_3 \\ \boldsymbol{p}_3^T\boldsymbol{p}_1 = 0 & \boldsymbol{p}_3^T\boldsymbol{p}_3 = \boldsymbol{p}_1^T\boldsymbol{p}_1 \end{array} \tag{5.29}$$

where, $\boldsymbol{p}_i$ refers to the elements of $\boldsymbol{p}$ corresponding to the i-th column of $\boldsymbol{P}$. These constraints are used to form a function $\boldsymbol{\zeta}(\cdot) : \mathbb{R}^{12} \rightarrow \mathbb{R}^6$ and the procedure of Section 5.2.3 is used.

5.2.5 Point Triangulation

Point triangulation is the problem of estimating the 3D position of a point given its position in various images of the scene. The data in this case consists of the pose of the camera $\boldsymbol{P}_i = [\boldsymbol{R}_i\ \ \boldsymbol{t}_i]$ and the 2D image position $\boldsymbol{x}_i = [x_{i1}\ \ x_{i2}]$. The 3D position of the point $\boldsymbol{X}$ needs to be estimated. The constraint can be expressed as

$$\begin{bmatrix} \boldsymbol{x}_i \\ 1 \end{bmatrix} \propto \boldsymbol{R}_i\boldsymbol{X} + \boldsymbol{t}_i . \tag{5.30}$$

Like before, this relation holds exactly only for ideal data. In this case, it can be rewritten as

$$\begin{bmatrix} \boldsymbol{r}_{i1o}^T - x_{i1o}\boldsymbol{r}_{i3o}^T & t_{i1o} - x_{i1o}t_{i3o} \\ \boldsymbol{r}_{i2o}^T - x_{i2o}\boldsymbol{r}_{i3o}^T & t_{i2o} - x_{i2o}t_{i3o} \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_o \\ 1 \end{bmatrix} = \Phi(\boldsymbol{x}_{io}, \boldsymbol{P}_{io})\boldsymbol{\theta}_o = \boldsymbol{0} \ . \tag{5.31}$$

In general, $\boldsymbol{r}_{ij}^T$ is the j-th row of $\boldsymbol{R}_i$ and t_{ij} is the j-th element of $\boldsymbol{t}_i$ and $\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{X}^T & 1\end{bmatrix}^T$. Now, the data vector consists of the pose and the image location and the nonlinear function Φ is given by (5.31). The HEIV cost function is homogeneous and the algorithm finds $\hat{\boldsymbol{\theta}}$ under the condition that $\|\hat{\boldsymbol{\theta}}\| = 1$. After finding the optimal $\hat{\boldsymbol{\theta}}$, $\boldsymbol{X}$ can be recovered by dividing $\hat{\boldsymbol{\theta}}$ by its last element.

5.3 Camera Tracking System

The learning framework of [46] was used as the basis of the tracking system of [115]. This system will later be extended to handle multiple sources of information.

We assume the camera has been calibrated offline. For our system, the calibration was done using Tsai's algorithm and allowing radial distortion up to the sixth degree [128]. This ensures good pose estimation when the system is provided with the right correspondences. The operation of the system is divided into the *learning phase* and the *tracking phase*. The *learning phase* is initialized when 3D markers, whose locations are known, are detected in the image sequence and the pose is computed based on these markers. The markers of [142] are used for this purpose. At each frame distinctive point features are detected using the algorithm of [119]. The covariances of these points up to a global scale factor can be obtained from the intensity values of the image [67]. Points correspondences across frames are found by the Lucas-Kanade tracker [79].

Given correspondences across frames and the pose for each of these frames, the feature points are triangulated. We assume there are no outliers present in the data. If necessary, a robust estimator such as RANSAC [37] or pbM [115] can be used at this stage to remove the outliers. Outliers could be erroneous point correspondences or frames with incorrect pose estimates.

Once the markers are not visible in the image the *tracking phase* begins. Well

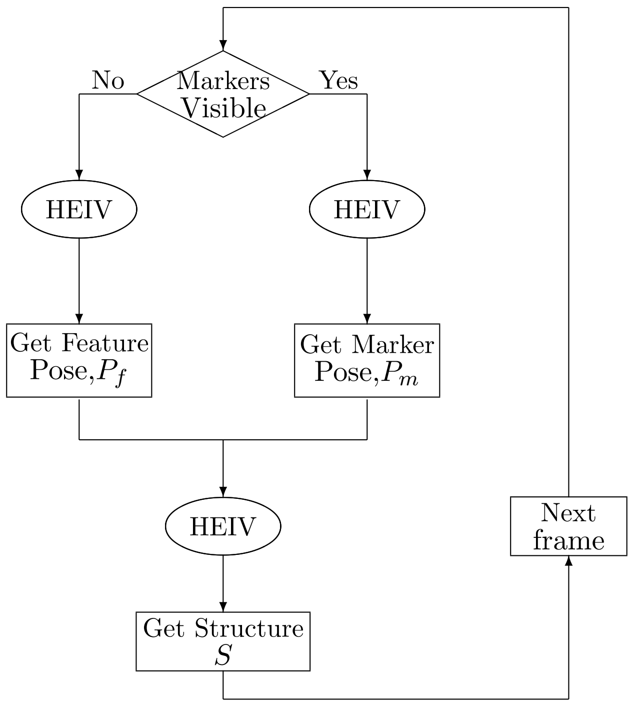


Figure 5.1: Outline of the tracking system.

reconstructed points of the learning stage are used to estimate the pose. We define well reconstructed points as points whose covariance matrices in 3D are such that the highest singular value is below some predefined threshold. Since the point have been triangulated through HEIV regression, their covariances are given by (5.21). New feature points are continually detected in the image sequence and triangulated so that at any given future frame a sufficient number of well-reconstructed points are available for pose computation. The complete system is graphically represented in Figure 5.1.

The covariance information of all the triangulated points and computed camera pose is retained and used in the HEIV estimation. In this manner points which are well localized in 3D get greater weight and in a similar manner pose estimates which are not good due to the planarity of the feature points etc. are neglected due to their high covariances.

The system discussed above was a modification of the method proposed in [46]. However, in [46], Levenberg-Marquardt (LM) regression was used instead of HEIV to handle the nonlinear regression. This led to a drift in the camera tracking as the camera moved away from the markers. The scene used to test the system can be seen in Figure

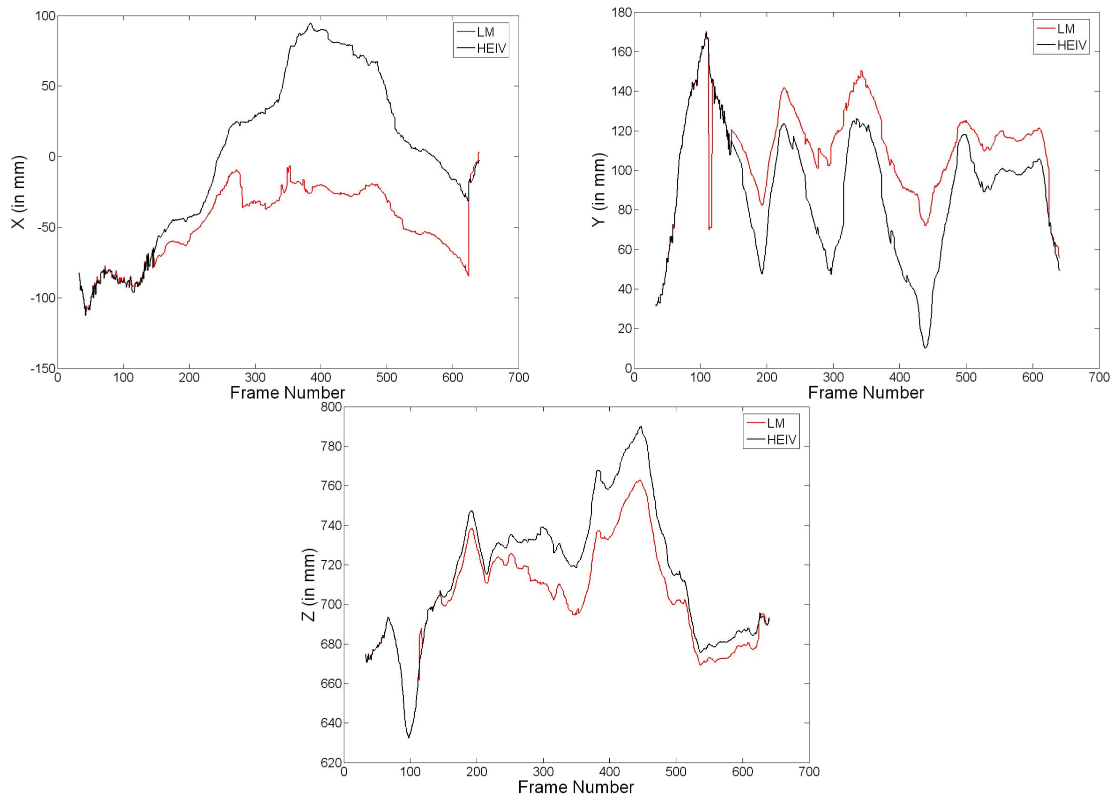


Figure 5.2: Camera paths for *workspace* scene.

5.3. The markers used during the learning phase can be seen at the top of the image.

The results of the system discussed in this section are compared with the results obtained by [46]. The image sequence consists of 641 frames. The markers come into view at Frame 32 and the learning phase begins. At frame 151 the markers are not visible anymore and the tracking phase begins. The camera now moves down towards the bottom of the scene and then moves back up. The markers come into view again at frame 624 and from this frame till the end the markers are again used for camera pose estimation.

The extent of drift suffered by each system can be judged by the difference in camera pose between fames 623 and frame 624. At frame 623, the markers are not visible and previously triangulated features are used for pose computation. However, in frame 624 the markers are used for pose estimation. Since the markers are easily identifiable in the image and their true 3D position is known, this pose is accurate. Since the camera motion is smooth, if no drift has occurred the difference between the two pose estimates

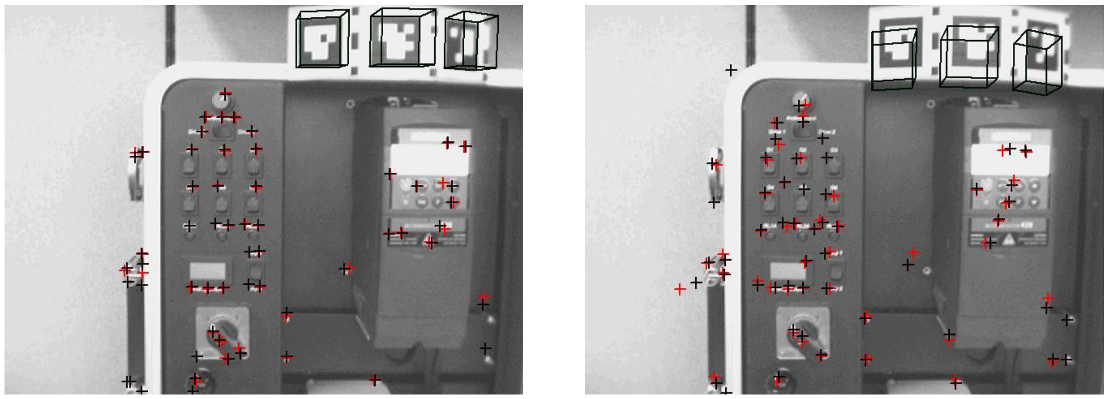


Figure 5.3: Comparison of camera tracking at frame 623 for the *workspace* experiment. The HEIV algorithm is used to get the pose and render the wireframe on the left and the LM algorithm is used on the right.

should not differ by much. To depict these ideas graphically, we use the pose to estimate the camera position for each frame. If the pose is given by $\boldsymbol{P} = [\boldsymbol{R}\ \ \boldsymbol{t}]$, the position of the camera center is given by $-\boldsymbol{R}^T\boldsymbol{t}$. The x, y and z coordinates of the camera center are plotted as functions of the frame number in Figure 5.2. The pose computed from triangulated features is used to render a wireframe around the markers at frame 623, just before the markers come back into view. These frames are shown in Figure 5.3.

In spite of the improvement offered by using HEIV regression, this system does not handle all the available information. Note that the pose for each camera is estimated separately. This ignores information available from the previous frame. The pose for the current frame cannot be very different from the pose at the previous frame. In a Kalman filtering based approach to SFM this continuity is enforced by the equations which model the dynamics of the system. To enforce this continuity, we use a gyroscope rather than explicitly model the dynamics using a filtering approach. The relation to Kalman filtering is further discussed in Section 5.4.1.

A gyroscope is a device for measuring or maintaining orientation, based on the principle of conservation of angular momentum. We use the Wireless InertiaCube3 gyroscope developed by InterSense [62]. The system provides three degrees-of-freedom orientation tracking. The angular estimates returned by the gyroscope have an uncertainty of 1° in yaw and 0.25° in pitch and roll. The gyroscope and the wireless receiver

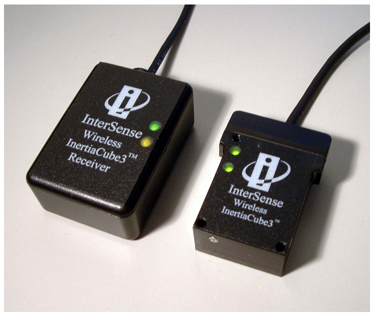


Figure 5.4: *Wireless Inertia Cube 3* and the wireless receiver.

are shown in Figure 5.4.

Based on the estimates returned by the gyroscope, the relative rotational motion between two frames is estimated. This can be used to obtain an estimate of the rotational part of the pose for the next frame. This way of updating also ensures that the pose for the next frame does not differ too much from the current pose. The rotational part of the pose obtained in this manner is noisy and translational part of the pose is unknown. In the next section we discuss a statistically balanced method for using the given world to image point correspondences to estimate the pose of the camera while also accounting for the noisy estimate given by the gyroscope.

5.4 The Modified HEIV Algorithm

In this section we derive an algorithm which is closely related to HEIV, but also accounts for other available information about the parameter vector. We derive the algorithm specifically for the problem of camera pose estimation given a set of world to image point correspondences *and* a noisy estimate of the rotation part of the pose. In our application this estimate of the rotation is supplied by the gyroscope. The method can also be used when only estimates of the translation are available, but this is not discussed here. Similarly, the method can be extended to other applications where some information about the parameter vector is available.

We are given a set of 3D world points and corresponding 2D image points which are organized into five dimensional vectors $\boldsymbol{z}_i,\ i = 1, \ldots, n$, as described in Section 5.2.4.

The covariance $\sigma^2 \boldsymbol{C}_{\boldsymbol{z}_i}$ of each $\boldsymbol{z}_i$ is known up to some common global scaling σ^2.

A noisy estimate $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ of the rotation part of the pose is also available. The matrix $\boldsymbol{R}$ is orthogonal and satisfies the constraints $\boldsymbol{R}\boldsymbol{R}^T = \mathbf{e_3}$. The elements of $\boldsymbol{R}$ can also be organized into a vector $\boldsymbol{r} \in \mathbb{R}^9$, which we shall refer to as the *rotation vector*. This estimate $\boldsymbol{r}$ differs from the true rotation vector $\boldsymbol{r}_o$ by additive noise whose covariance is given by $\boldsymbol{C}_{\boldsymbol{r}} \in \mathbb{R}^{9\times9}$ which is known up to the same scaling of σ^2. The orthogonality constraints on $\boldsymbol{R}$ give rise to 6 linearly independent nonlinear constraints on the vector $\boldsymbol{r}$ and the vector $\boldsymbol{r}$ has only three degrees of freedom. Therefore, $\boldsymbol{C}_{\boldsymbol{r}}$ is a 9×9, symmetric, positive-semi definite matrix whose rank is only three. By performing SVD on $\boldsymbol{C}_{\boldsymbol{r}}$ we obtain the 9×3 matrix $\boldsymbol{U}_{\boldsymbol{r}}$ and the 3×3 diagonal matrix $\boldsymbol{S}_{\boldsymbol{r}}$ such that $\boldsymbol{C}_{\boldsymbol{r}} = \boldsymbol{U}_{\boldsymbol{r}}\boldsymbol{S}_{\boldsymbol{r}}\boldsymbol{U}_{\boldsymbol{r}}^T$. The reason for doing this is explained later.

The true point matches, rotation and translation satisfy the *projection equations* given by (5.27). We denote these constraints between $\boldsymbol{z}_{io}$, $\boldsymbol{r}_o$ and the translation $\boldsymbol{t}_o$ as $\boldsymbol{f}(\boldsymbol{z}_{io}, \boldsymbol{r}_o, \boldsymbol{t}_o) = \mathbf{0}$. The projection equations can also be rewritten as

$$\boldsymbol{f}(\boldsymbol{z}_i, \boldsymbol{r}, \boldsymbol{t}) = \Phi(\boldsymbol{z}_i, \boldsymbol{r})\boldsymbol{\theta} \tag{5.32}$$

where, the unknown translation is contained in $\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{t}^T & 1\end{bmatrix}^T \in \mathbb{R}^4$ and $\Phi(\boldsymbol{z}_i, \boldsymbol{r}) \in \mathbb{R}^{2\times4}$ is a nonlinear function of the point match and rotation given by

$$\Phi(\boldsymbol{z}_i, \boldsymbol{r}) = \begin{bmatrix} 1 & 0 & -u_i & \boldsymbol{r}_1^T\boldsymbol{p}_i^3 - u_i\boldsymbol{r}_3^T\boldsymbol{p}_i^3 \\ 0 & 1 & -v_i & \boldsymbol{r}_2^T\boldsymbol{p}_i^3 - v_i\boldsymbol{r}_3^T\boldsymbol{p}_i^3 \end{bmatrix}. \tag{5.33}$$

where, $\boldsymbol{r}_j^T$ is the j-th row of the rotation matrix, $\boldsymbol{p}_i^3$ is the world position of the i-th point correspondence and (u_i, v_i) is the image position of the i-th point.

The aim is to obtain an optimal estimate of the translation $\boldsymbol{t} \in \mathbb{R}^3$ *and* optimally corrected values of $\boldsymbol{z}_i$ and $\boldsymbol{r}$. The vector $\boldsymbol{r}$ is involved in all the n functional constraints and appears in all the carrier vectors. This correlation between the n carrier vectors implies that the simple HEIV algorithm of Section 5.2 is no longer valid.

We proceed by minimizing the cost function

$$\mathcal{J} \quad = \quad \frac{1}{2}\sum_i(\hat{\boldsymbol{z}}_i - \boldsymbol{z}_i)^T\boldsymbol{C}_{\boldsymbol{z}_i}^{+}(\hat{\boldsymbol{z}}_i - \boldsymbol{z}_i) + \frac{1}{2}(\hat{\boldsymbol{r}} - \boldsymbol{r})^T\boldsymbol{C}_{\boldsymbol{r}}^{+}(\hat{\boldsymbol{r}} - \boldsymbol{r}) \tag{5.34}$$

while satisfying the constraints,

$$\boldsymbol{f}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}}) = \boldsymbol{0}, \;\; i = 1, \ldots, n \tag{5.35}$$

In our application, the matrices $\boldsymbol{C}_{\boldsymbol{z}_i}$ are full rank and the pseudo-inverse $\boldsymbol{C}_{zi}^{+}$ is equivalent to the inverse $\boldsymbol{C}_i^{-1}$. In a general case it is possible for $\boldsymbol{C}_{\boldsymbol{z}_i}$ to be rank-deficient.

5.4.1 Relationship to Kalman Filtering Approaches

The cost function being minimized above is a likelihood function, if the additive noise is Gaussian. In this case, the minima of the cost function is a *maximum likelihood* (ML) estimate. Kalman filtering also tries to obtain a ML estimate under the assumption of Gaussian noise. The major difference between the approach that we propose and Kalman filter based approaches to structure-from-motion [5, 18] is the difference in the model used for the dynamics. In [18], the rotational and translational velocity of the camera are taken to be part of the state vector. The *a priori* estimate of velocity for the next frame is taken to be the *a posteriori* estimate from the previous frame. Any change in the velocity i.e. acceleration is modeled as white noise. The validity of this assumption can only be justified by the results obtained in practice.

In our system, we do not explicitly model the velocity or the acceleration. Rather, we use a gyroscope to update our position estimates and get *a priori* estimates of the next camera pose. However, our cost function does not place any constraints on the manner in which these *a priori* constraints are obtained. For example, rather than using the gyroscope, the velocity of the camera can be taken as part of the state. In this case, an *a posteriori* estimate of this velocity is always available and this estimate can be used to update the current camera position to obtain an *a priori* camera pose estimate for the next frame.

The advantage that our method offers over a Kalman filter is that it handles the nonlinearities in the projection equations. Kalman filters, are theoretically valid only for linear systems. In practice, extended Kalman filters are used to handle the nonlinearities, but the drawbacks of extended Kalman filters have been well documented [64]. More complex methods such as Unscented Kalman filters [64] have been proposed, but the complexity of these algorithms restricts them to low dimensional problems. Our system is closest to the Iterated Extended Kalman Filter [45], but our function optimization is not a Gauss-Newton method.

5.4.2 Derivation

Introducing Lagrange multipliers $\boldsymbol{\eta}_i \in \mathbb{R}^2$, the cost function becomes

$$\mathcal{J} = \frac{1}{2}\sum_{i=1}^{n}(\hat{\boldsymbol{z}}_i - \boldsymbol{z}_i)^T \boldsymbol{C}^{+}_{\boldsymbol{z}_i}(\hat{\boldsymbol{z}}_i - \boldsymbol{z}_i) + \frac{1}{2}(\hat{\boldsymbol{r}} - \boldsymbol{r})^T \boldsymbol{C}^{+}_{\boldsymbol{r}}(\hat{\boldsymbol{r}} - \boldsymbol{r}) + \sum_{i=1}^{n}\boldsymbol{\eta}_i^T \boldsymbol{f}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}})\ . \tag{5.36}$$

At the minima, the Jacobians of $\mathcal{J}$ with respect to $\hat{\boldsymbol{z}}_i$, $\hat{\boldsymbol{r}}$ and $\hat{\boldsymbol{\theta}}$ should satisfy

$$\boldsymbol{J}_{\mathcal{J}|\boldsymbol{z}_i} = \boldsymbol{C}^{+}_{\boldsymbol{z}_i}(\hat{\boldsymbol{z}}_i - \boldsymbol{z}_i) + \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}})\boldsymbol{\eta}_i = \boldsymbol{0} \tag{5.37}$$

$$\boldsymbol{J}_{\mathcal{J}|\boldsymbol{r}} = \boldsymbol{C}^{+}_{\boldsymbol{r}}(\hat{\boldsymbol{r}} - \boldsymbol{r}) + \textstyle\sum_{j=1}^{n}\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{r}}(\hat{\boldsymbol{z}}_j, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}})\boldsymbol{\eta}_j = \boldsymbol{0} \tag{5.38}$$

$$\boldsymbol{J}_{\mathcal{J}|\boldsymbol{\theta}} = \textstyle\sum_{i=1}^{n}\Phi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}})^T\boldsymbol{\eta}_i = \boldsymbol{0}\ . \tag{5.39}$$

The equations (5.37) and (5.38) can be rewritten as

$$\hat{\boldsymbol{z}}_i = \boldsymbol{z}_i - \boldsymbol{C}_{\boldsymbol{z}_i}\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}})\boldsymbol{\eta}_i \tag{5.40}$$

$$\hat{\boldsymbol{r}} = \boldsymbol{r} - \boldsymbol{C}_{\boldsymbol{r}}\textstyle\sum_{j=1}^{n}\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{r}}(\hat{\boldsymbol{z}}_j, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}})\boldsymbol{\eta}_j \tag{5.41}$$

Like in the original HEIV, let $\check{\boldsymbol{z}}_i$ and $\check{\boldsymbol{r}}$, be available estimates. Initially, we assume $\check{\boldsymbol{z}}_i = \boldsymbol{z}_i$ and $\check{\boldsymbol{r}} = \boldsymbol{r}$. The first order Taylor expansion of $\boldsymbol{f}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}})$ around $\check{\boldsymbol{z}}_i$ and $\check{\boldsymbol{r}}$ gives

$$\boldsymbol{f}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}}) = \boldsymbol{f}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}}) + \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})^T(\hat{\boldsymbol{z}}_i - \check{\boldsymbol{z}}_i) + \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{r}}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})^T(\hat{\boldsymbol{r}} - \check{\boldsymbol{r}}) = \boldsymbol{0}. \tag{5.42}$$

We assume that the Jacobians $\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}})$ and $\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{r}}(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}})$ do not change much when computed at $\check{\boldsymbol{z}}_i$ and $\check{\boldsymbol{r}}$ instead of $\hat{\boldsymbol{z}}_i$ and $\hat{\boldsymbol{r}}$. This assumption is reasonable since the estimates $\check{\boldsymbol{z}}_i$ and $\check{\boldsymbol{r}}$ are quite close to $\hat{\boldsymbol{z}}_i$ and $\hat{\boldsymbol{r}}$. The equation (5.40) and (5.41) can be rewritten as

$$\hat{\boldsymbol{z}}_i - \check{\boldsymbol{z}}_i = \boldsymbol{z}_i - \check{\boldsymbol{z}}_i - \boldsymbol{C}_{\boldsymbol{z}_i}\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})\boldsymbol{\eta}_i \tag{5.43}$$

$$\hat{\boldsymbol{r}} - \check{\boldsymbol{r}} = \boldsymbol{r} - \check{\boldsymbol{r}} - \boldsymbol{C}_{\boldsymbol{r}}\textstyle\sum_{j=1}^{n}\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{r}}(\check{\boldsymbol{z}}_j, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})\boldsymbol{\eta}_j \tag{5.44}$$

Using the above expressions for $\hat{\boldsymbol{z}}_i - \check{\boldsymbol{z}}_i$ and $\hat{\boldsymbol{r}} - \check{\boldsymbol{r}}$ and recognizing the first order Taylor series expansion of $\boldsymbol{f}(\boldsymbol{z}_i, \boldsymbol{r}, \hat{\boldsymbol{t}})$ around $\check{\boldsymbol{z}}_i$ and $\check{\boldsymbol{r}}$, the equation (5.42) becomes

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{z}_i, \boldsymbol{r}, \hat{\boldsymbol{t}}) &= \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})^T \boldsymbol{C}_{\boldsymbol{z}_i} \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}}) \boldsymbol{\eta}_i \\ &\quad + \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{r}}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})^T \boldsymbol{C}_{\boldsymbol{r}} \sum_{j=1}^{n} \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{r}}(\check{\boldsymbol{z}}_j, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}}) \boldsymbol{\eta}_j \end{aligned} \tag{5.45}$$

Recall that $\boldsymbol{C}_{\boldsymbol{r}} = \boldsymbol{U}_{\boldsymbol{r}} \boldsymbol{S}_{\boldsymbol{r}} \boldsymbol{U}_{\boldsymbol{r}}^T$. We define

$$\boldsymbol{\Sigma}_i \triangleq \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})^T \boldsymbol{C}_{\boldsymbol{z}_i} \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}}) \qquad \boldsymbol{X}_i \triangleq \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{r}}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})^T \boldsymbol{U}_{\boldsymbol{r}} \ . \tag{5.46}$$

In terms of these matrices, equation (5.45) becomes

$$\boldsymbol{f}(\boldsymbol{z}_i, \boldsymbol{r}, \hat{\boldsymbol{t}}) = \boldsymbol{\Sigma}_i \boldsymbol{\eta}_i + \boldsymbol{X}_i \boldsymbol{S}_{\boldsymbol{r}} \sum_{j=1}^{n} \boldsymbol{X}_j^T \boldsymbol{\eta}_j \tag{5.47}$$

As i varies from 1 to n we get n equations of the form of (5.47). This complete system of equation can be represented in a single matrix equation as,

$$(\boldsymbol{A} + \boldsymbol{X} \boldsymbol{S}_{\boldsymbol{r}} \boldsymbol{X}^T) \boldsymbol{\eta} = \boldsymbol{b} \tag{5.48}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Sigma}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\Sigma}_n \end{bmatrix} \quad \boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \vdots \\ \boldsymbol{X}_n \end{bmatrix} \quad \boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \\ \vdots \\ \boldsymbol{\eta}_n \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{z}_1, \boldsymbol{r}, \hat{\boldsymbol{t}}) \\ \boldsymbol{f}(\boldsymbol{z}_2, \boldsymbol{r}, \hat{\boldsymbol{t}}) \\ \vdots \\ \boldsymbol{f}(\boldsymbol{z}_n, \boldsymbol{r}, \hat{\boldsymbol{t}}) \end{bmatrix} \tag{5.49}$$

where $\boldsymbol{A} \in \mathbb{R}^{2n \times 2n}$, $\boldsymbol{X} \in \mathbb{R}^{2n \times 3}$ and $\boldsymbol{\eta}, \boldsymbol{b} \in \mathbb{R}^{2n}$. The Lagrange multipliers $\boldsymbol{\eta}$, are given by,

$$\boldsymbol{\eta} = (\boldsymbol{A} + \boldsymbol{X} \boldsymbol{S}_{\boldsymbol{r}} \boldsymbol{X}^T)^{-1} \boldsymbol{b} \ . \tag{5.50}$$

Using the Sherman-Morrison-Woodbury (SWM) formula for computing the matrix inverse,

$$\boldsymbol{\eta} = (\boldsymbol{A}^{-1} - \boldsymbol{A}^{-1} \boldsymbol{X} \boldsymbol{M} \boldsymbol{X}^T \boldsymbol{A}^{-1}) \boldsymbol{b} \tag{5.51}$$

where,

$$\boldsymbol{M} = (\boldsymbol{S}_{\boldsymbol{r}}^{-1} + \boldsymbol{X}^T \boldsymbol{A}^{-1} \boldsymbol{X})^{-1} \ . \tag{5.52}$$

Since, $\boldsymbol{A}$ is a block diagonal matrix, its inverse can be computed simply by inverting each 2×2 matrix $\boldsymbol{\Sigma}_i$. The only other matrix that needs to be inverted is $\boldsymbol{S}_{\boldsymbol{r}}^{-1} + \boldsymbol{X}^T \boldsymbol{A}^{-1} \boldsymbol{X}$, which is a 3×3 matrix. Therefore, the SMW formula drastically reduces the complexity of inverting the $2n \times 2n$ matrix $\boldsymbol{A} + \boldsymbol{X} \boldsymbol{S}_{\boldsymbol{r}} \boldsymbol{X}^T$. The Lagrange multipliers are given by

$$\boldsymbol{\eta}_i = \boldsymbol{\Sigma}_i^+ \boldsymbol{f}(\boldsymbol{z}_i, \boldsymbol{r}, \hat{\boldsymbol{t}}) - \boldsymbol{\Sigma}_i^+ \boldsymbol{X}_i \boldsymbol{M} \sum_{j=1}^{n} \boldsymbol{X}_j^T \boldsymbol{\Sigma}_j^+ \boldsymbol{f}(\boldsymbol{z}_i, \boldsymbol{r}, \hat{\boldsymbol{t}}) \tag{5.53}$$

This is the reason for replacing $\boldsymbol{C}_{\boldsymbol{r}}$ by its SVD. Without this change, the SWM formula would not have been applicable and the matrix would have to be explicitly inverted. This is required only when $\boldsymbol{C}_{\boldsymbol{r}}$ is singular. If this is not the case then the SVD is not necessary.

At the minima we also require the Jacobian with respect to $\hat{\boldsymbol{\theta}}$ to be zero. This gives

$$\boldsymbol{J}_{\mathcal{J}|\boldsymbol{\theta}} \;=\; \sum_{i=1}^{n} \Phi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}})^T \boldsymbol{\eta}_i = \sum_{i=1}^{n} (\boldsymbol{\eta}_i \otimes \mathbf{I}_p)^T \varphi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}) = 0 \tag{5.54}$$

where, $\otimes$ is the Kronecker product [52] and $\varphi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}) = \mathbf{vec}(\Phi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}})^T)$. The first order expansion of $\varphi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}})$ around $\check{\boldsymbol{z}}_i$ and $\check{\boldsymbol{r}}$ gives

$$\varphi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}) = \varphi(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) + \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})^T (\hat{\boldsymbol{z}}_i - \check{\boldsymbol{z}}_i) + \boldsymbol{J}_{\varphi|\boldsymbol{r}}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})^T (\hat{\boldsymbol{r}} - \check{\boldsymbol{r}}) \tag{5.55}$$

Substituting for $\hat{\boldsymbol{z}}_i - \check{\boldsymbol{z}}_i$ and $\hat{\boldsymbol{r}} - \check{\boldsymbol{r}}$, from (5.43) and (5.44) in the above equation and recognizing the first order Taylor series expansion of $\varphi(\boldsymbol{z}_i, \boldsymbol{r})$ about $\check{\boldsymbol{z}}_i$, $\check{\boldsymbol{r}}$

$$\varphi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}) = \varphi(\boldsymbol{z}_i, \boldsymbol{r}) - \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})^T \boldsymbol{C}_{\boldsymbol{z}_i} \boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) \boldsymbol{\eta}_i - \boldsymbol{J}_{\varphi|\boldsymbol{r}}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})^T \boldsymbol{C}_{\boldsymbol{r}} \sum_{j=1}^{n} \boldsymbol{J}_{\varphi|\boldsymbol{r}}(\check{\boldsymbol{z}}_j, \check{\boldsymbol{r}}) \boldsymbol{\eta}_j \tag{5.56}$$

Using the definition of the Kronecker product for $\check{\boldsymbol{z}}_i$ and $\check{\boldsymbol{r}}$

$$\boldsymbol{f}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}}) = \Phi(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) \hat{\boldsymbol{\theta}} = (\mathbf{e_m} \otimes \hat{\boldsymbol{\theta}})^T \varphi(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) \; . \tag{5.57}$$

Evaluating Jacobians with respect to $\check{\boldsymbol{z}}_i$ and $\check{\boldsymbol{r}}$ on both sides of the above equations we get

$$\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}}) = \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})(\mathbf{e_m} \otimes \hat{\boldsymbol{\theta}}) \tag{5.58}$$

$$\boldsymbol{J}_{\boldsymbol{f}|\boldsymbol{r}}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}}) = \boldsymbol{J}_{\varphi|\boldsymbol{r}}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}, \hat{\boldsymbol{t}})(\mathbf{e_m} \otimes \hat{\boldsymbol{\theta}}) \tag{5.59}$$

The equation (5.56) now becomes

$$\begin{aligned}\varphi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}) = \varphi(\boldsymbol{z}_i, \boldsymbol{r}) - \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})^T \boldsymbol{C}_{\boldsymbol{z}_i} \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})(\boldsymbol{\eta}_i \otimes \mathbf{e_p})\hat{\boldsymbol{\theta}} \\ -\boldsymbol{J}_{\varphi|\boldsymbol{r}}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})^T \boldsymbol{C}_{\boldsymbol{r}} \sum_{j=1}^{n} \boldsymbol{J}_{\varphi|\boldsymbol{r}}(\check{\boldsymbol{z}}_j, \check{\boldsymbol{r}})(\boldsymbol{\eta}_j \otimes \mathbf{e_p})\hat{\boldsymbol{\theta}}\end{aligned} \tag{5.60}$$

The covariance of $\varphi(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})$ approximated by error propagation in $\check{\boldsymbol{z}}_i$ is

$$\boldsymbol{C}_{\varphi}(\check{\boldsymbol{z}}_i) = \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})^T \boldsymbol{C}_{\boldsymbol{z}_i} \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) \ . \tag{5.61}$$

Similarly the cross covariance between $\varphi(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})$ and $\varphi(\check{\boldsymbol{z}}_j, \check{\boldsymbol{r}})$ by error propagation in $\check{\boldsymbol{r}}$ is

$$\boldsymbol{C}_{\varphi}(\check{\boldsymbol{r}}) = \boldsymbol{U}_{\varphi}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) \boldsymbol{U}_{\varphi}(\check{\boldsymbol{z}}_j, \check{\boldsymbol{r}})^T = \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})^T \boldsymbol{C}_{\boldsymbol{r}} \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) \tag{5.62}$$

where $\boldsymbol{U}_{\varphi}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) = \boldsymbol{J}_{\varphi|\boldsymbol{z}_i}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}})^T \boldsymbol{C}_{\boldsymbol{r}}^{1/2}$. In terms of these matrices, (5.60) becomes

$$\varphi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}}) = \varphi(\boldsymbol{z}_i, \boldsymbol{r}) - \boldsymbol{C}_{\varphi}(\check{\boldsymbol{z}}_i)(\boldsymbol{\eta}_i \otimes \mathbf{e_p})\hat{\boldsymbol{\theta}} - \boldsymbol{U}_{\varphi}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) \sum_{j=1}^{n} \boldsymbol{U}_{\varphi}(\check{\boldsymbol{z}}_j, \check{\boldsymbol{r}})^T (\boldsymbol{\eta}_j \otimes \mathbf{e_p})\hat{\boldsymbol{\theta}} \ . \tag{5.63}$$

Substituting for $\varphi(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{r}})$ in (5.54), and reorganizing the terms we get

$$\begin{aligned}\sum_{i=1}^{n} (\boldsymbol{\eta}_i \otimes \mathbf{I}_p)^T \varphi(\boldsymbol{z}_i, \boldsymbol{r}) &= \sum_{i=1}^{n} (\boldsymbol{\eta}_i \otimes \mathbf{I}_p)^T \boldsymbol{C}_{\varphi}(\check{\boldsymbol{z}}_i)(\boldsymbol{\eta}_i \otimes \mathbf{I}_p)\hat{\boldsymbol{\theta}} \\ &\quad + \sum_{i=1}^{n} \sum_{j=1}^{n} (\boldsymbol{\eta}_i \otimes \mathbf{I}_p)^T \boldsymbol{U}_{\varphi}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) \boldsymbol{U}_{\varphi}(\check{\boldsymbol{z}}_j, \check{\boldsymbol{r}})^T (\boldsymbol{\eta}_j \otimes \mathbf{e_p})\hat{\boldsymbol{\theta}}\end{aligned} \tag{5.64}$$

By the properties of the Kronecker product, the left side of the above equation is equivalent to $\sum_{i=1}^{n} \Phi(\boldsymbol{z}_i, \boldsymbol{r})^T \boldsymbol{\eta}_i$. Substituting for the Lagrange multipliers using (5.53) we get

$$\begin{aligned}\sum_{i=1}^{n} \Phi(\boldsymbol{z}_i, \boldsymbol{r})^T \boldsymbol{\eta}_i &= \sum_{i=1}^{n} \Phi(\boldsymbol{z}_i, \boldsymbol{r})^T \boldsymbol{\Sigma}_i^{+} \Phi(\boldsymbol{z}_i, \boldsymbol{r}) \hat{\boldsymbol{\Theta}} \\ &\quad - \sum_{i=1}^{n} \sum_{j=1}^{n} \Phi(\boldsymbol{z}_i, \boldsymbol{r})^T \boldsymbol{\Sigma}_i \boldsymbol{X}_i \boldsymbol{M} \boldsymbol{X}_j^T \boldsymbol{\Sigma}_j^{+} \Phi(\boldsymbol{z}_j, \boldsymbol{r}) \hat{\boldsymbol{\Theta}}\end{aligned} \tag{5.65}$$

where $\boldsymbol{M}$ is given by (5.52). The equation (5.64) can be written like the HEIV equation (5.14)

$$\boldsymbol{S}(\hat{\boldsymbol{\theta}})\hat{\boldsymbol{\theta}} = \boldsymbol{C}(\hat{\boldsymbol{\theta}})\hat{\boldsymbol{\theta}} \tag{5.66}$$

where, $\boldsymbol{S}(\hat{\boldsymbol{\theta}})$ is the scatter matrix given by

$$\begin{aligned}\boldsymbol{S}(\hat{\boldsymbol{\theta}}) &= \sum_{i=1}^{n} \Phi(\boldsymbol{z}_i, \boldsymbol{r})^T \boldsymbol{\Sigma}_i^{+} \Phi(\boldsymbol{z}_i, \boldsymbol{r}) \\ & -\sum_{i=1}^{n}\sum_{j=1}^{n} \Phi(\boldsymbol{z}_i, \boldsymbol{r})^T \boldsymbol{\Sigma}_i \boldsymbol{X}_i \boldsymbol{M} \boldsymbol{X}_j^T \boldsymbol{\Sigma}_j^{+} \Phi(\boldsymbol{z}_j, \boldsymbol{r}) \end{aligned} \tag{5.67}$$

and $\mathbf{C}(\hat{\boldsymbol{\theta}})$ is the covariance matrix given by

$$\begin{aligned}\boldsymbol{C}(\hat{\boldsymbol{\theta}}) &= \sum_{i=1}^{n} (\boldsymbol{\eta}_i \otimes \mathbf{I}_p)^T \boldsymbol{C}_{\varphi}(\check{\boldsymbol{z}}_i)(\boldsymbol{\eta}_i^T \otimes \mathbf{I}_p) \\ & +\sum_{i=1}^{n}\sum_{j=1}^{n} (\boldsymbol{\eta}_i \otimes \mathbf{I}_p)^T \boldsymbol{U}_{\varphi}(\check{\boldsymbol{z}}_i, \check{\boldsymbol{r}}) \boldsymbol{U}_{\varphi}(\check{\boldsymbol{z}}_j, \check{\boldsymbol{r}})^T (\boldsymbol{\eta}_j \otimes \mathbf{e_p})\ . \end{aligned} \tag{5.68}$$

The expressions should be compared with the scatter and covariance matrices defined for the original HEIV by (5.9) and (5.10). The first terms involving summation over i exist but the cross-terms involving summation over i and j are not present in (5.9) and (5.10). These appear now due to the correlation introduced by $\boldsymbol{r}$.

Like in standard HEIV, due to the dependence of $\mathbf{S}(\hat{\boldsymbol{\theta}})$ and $\mathbf{C}(\hat{\boldsymbol{\theta}})$ on $\hat{\boldsymbol{\theta}}$, (5.66) cannot be solved for $\hat{\boldsymbol{\theta}}$ directly. The generalized singular value decomposition (GSVD) problem of (5.66) is solved iteratively. Starting from an initial estimate of $\hat{\boldsymbol{\theta}}$, the scatter matrix and covariance matrix are estimated. In practice, The GSVD problem is solved by converting it into a generalized eigenvalue problem in terms of the matrix square roots of $\boldsymbol{S}(\hat{\boldsymbol{\theta}})$ and $\boldsymbol{C}(\hat{\boldsymbol{\theta}})$. The generalized eigenvalue problem is solved to find a better estimate of $\hat{\boldsymbol{\theta}}$. The current definition of the scatter matrix makes this difficult since it cannot be easily factored into its matrix square root. Therefore, we redefine the matrices as

$$\bar{\boldsymbol{S}}(\hat{\boldsymbol{\theta}}) = \textstyle\sum_{i=1}^{n} \Phi(\boldsymbol{z}_i, \boldsymbol{r})^T \boldsymbol{\Sigma}_i^{+} \Phi(\boldsymbol{z}_i, \boldsymbol{r}) \tag{5.69}$$

$$\bar{\boldsymbol{C}}(\hat{\boldsymbol{\theta}}) = \boldsymbol{C}(\hat{\boldsymbol{\theta}}) + \textstyle\sum_{i=1}^{n}\sum_{j=1}^{n} \Phi(\boldsymbol{z}_i, \boldsymbol{r})^T \boldsymbol{\Sigma}_i \boldsymbol{X}_i \boldsymbol{M} \boldsymbol{X}_j^T \boldsymbol{\Sigma}_j^{+} \Phi(\boldsymbol{z}_j, \boldsymbol{r}) \tag{5.70}$$

Similar to (5.66), at the k-th iteration, we have the equation,

$$\bar{\boldsymbol{S}}(\hat{\boldsymbol{\theta}}^{[k]})\hat{\boldsymbol{\theta}}^{[k+1]} = \bar{\boldsymbol{C}}(\hat{\boldsymbol{\theta}}^{[k]})\hat{\boldsymbol{\theta}}^{[k+1]} \tag{5.71}$$

However, now is easy to take the matrix square root of $\bar{\boldsymbol{S}}(\hat{\boldsymbol{\theta}})$ and hence (5.71) can be solved by solving a generalized eigenvalue problem.

5.4.3 Statistical Properties

In our system, we shall require the covariance of the pose estimates. This is done so that pose estimates which are well localized (have low covariance) are given more weight than pose estimates with high covariances. For this purpose we require the *a posteriori* covariances of the pose estimates returned by the algorithm of the previous section. The *a posteriori* covariance of $\hat{\boldsymbol{r}}$ is given by

$$\boldsymbol{C}_{\hat{\boldsymbol{r}}} = \boldsymbol{C}_{\boldsymbol{r}} - \boldsymbol{U}_{\boldsymbol{r}}\boldsymbol{S}_{\boldsymbol{r}}(\boldsymbol{T} - \boldsymbol{TMT})\boldsymbol{S}_{\boldsymbol{r}}\boldsymbol{U}_{\boldsymbol{r}} \tag{5.72}$$

where, $\boldsymbol{T} = \boldsymbol{X}^T\boldsymbol{A}^{-1}\boldsymbol{X}$. The *a posteriori* covariance of $\hat{\boldsymbol{\theta}}$ is given by

$$\boldsymbol{C}_{\hat{\boldsymbol{\theta}}} = (\boldsymbol{S}(\hat{\boldsymbol{\theta}}) - \boldsymbol{C}(\hat{\boldsymbol{\theta}}))^{+} \tag{5.73}$$

5.5 Experimental Results

The above algorithm for fusing the rotation estimates supplied by the gyroscope, with the image data was integrated into the camera tracking system discussed in Section 5.3. The tracking phase begins from the first frame in which the markers are visible. For this frame, the pose is computed using standard HEIV pose computation.

For each of the further frames, we have an estimate of the pose of the previous frame. The gyroscope is used to get an estimate of the relative rotation between the previous frame and the current frame. Given the pose of the previous frame and the relative rotation, we get a estimate for the rotation part of the current pose by matrix multiplication.

Note, that the gyroscope returns estimates of the actual orientation. However, the gyroscope also suffers from drift which increases with time. Therefore, for longer sequences the estimates of the absolute orientation returned by the gyroscope deteriorate rapidly. To overcome this we compare the current orientation estimate of the gyroscope with the previous orientation to obtain the relative rotation between the two frames. This method has the further advantage that it is not necessary to know the position and orientation of the gyroscope with respect to the camera. As long as the gyroscope

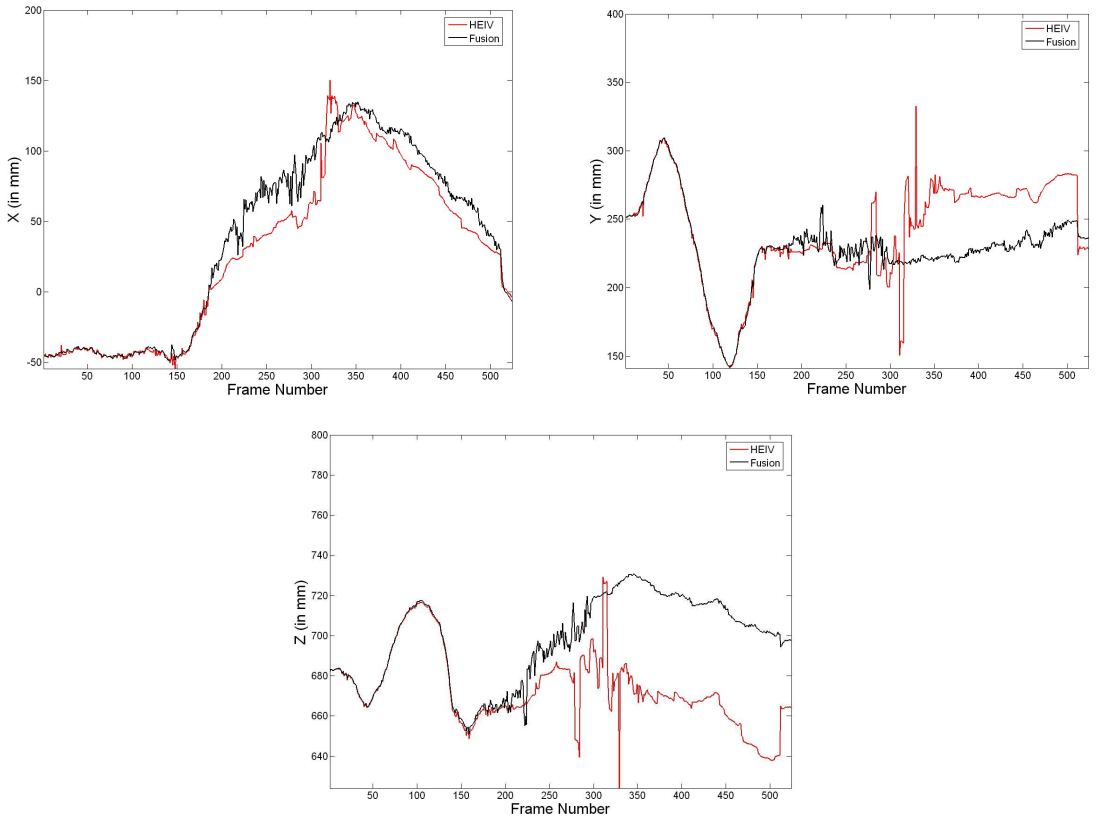


Figure 5.5: Camera paths for experiment 1. The results of the simple HEIV algorithm are compared with our fusion algorithm. The discontinuity in the position estimates when the markers come back into view at frame 520 is an indication of the drift suffered by the system. The fusion based system clearly suffers from lesser drift.

it attached to the camera rigidly, the constant transformation between the camera and gyroscope systems does not affect the relative rotation estimates.

In the *learning phase*, given the covariances of the previous orientation and the relative rotation returned by the gyroscope, the covariance of the *a priori* orientation estimate can be obtained through error propagation. The markers are detected in the image, and as their world position is known we know a sufficient number of image to world correspondences. This data is combined using the new HEIV algorithm of Section 5.4 to obtain the *a posteriori* pose of the camera.

The pose estimation works similarly in the *tracking phase*. The only difference is that the image to world point correspondences are obtained by tracking scene features. We compare the results of this system with those obtained by the system [115] discussed

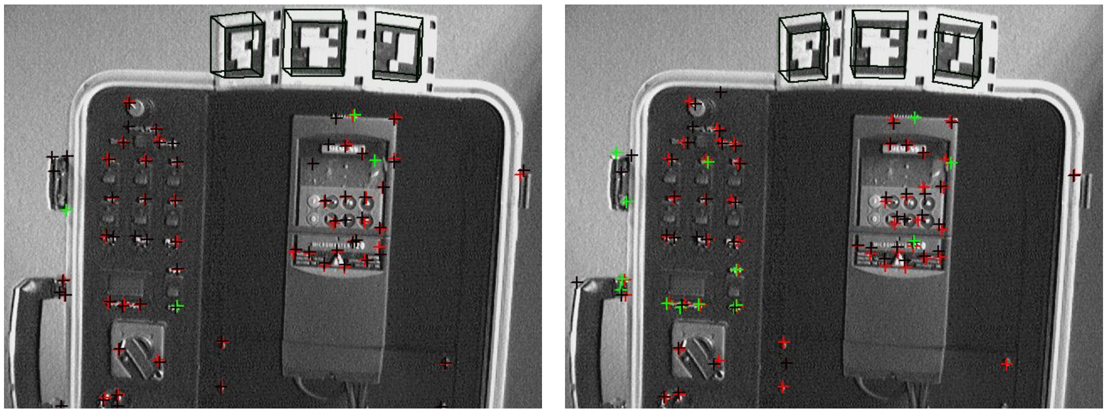


Figure 5.6: Comparison of camera tracking at frame 519 of *experiment one*. On the left the fusion algorithm is used to get the pose and render the wireframe while the HEIV algorithm is used on the right.

in Section 5.3.

The algorithms were run on a single processor Pentium III, 2.66 GHz machine and processed the sequences at 15-25 frames per second (fps). Although, this is not quite frame rate, we believe that with a the system can be made to run at frame rate by improving various sections of the code. Also, faster computers are available on the market and faster processors should considerably improve frame rate.

The *first experiment* is carried out on the scene as before, although this time we also use the gyroscope while capturing the frame sequence. The sequence is 524 frames long and the camera is moved quickly to give the system less time to localize new features. The pose of the camera for each frame is used to estimate the position of the camera center. The coordinates of the camera center are plotted versus the frame numbers to get an idea of the camera motion. The camera paths returned by the two systems are shown in Figure 5.5. The markers are kept in view for the first 150 frames to allow localization of various scene features. Due to this both the HEIV and fusion algorithms give smooth estimates of the camera position for the first 150 frames as can be seen in the three plots of the camera position versus frame number.

After this the camera is moved away from the markers so that the scene features are used for pose computation. This makes the position estimates more noisy and the system also starts to suffer from drift. Note, that as soon as the markers go out of view

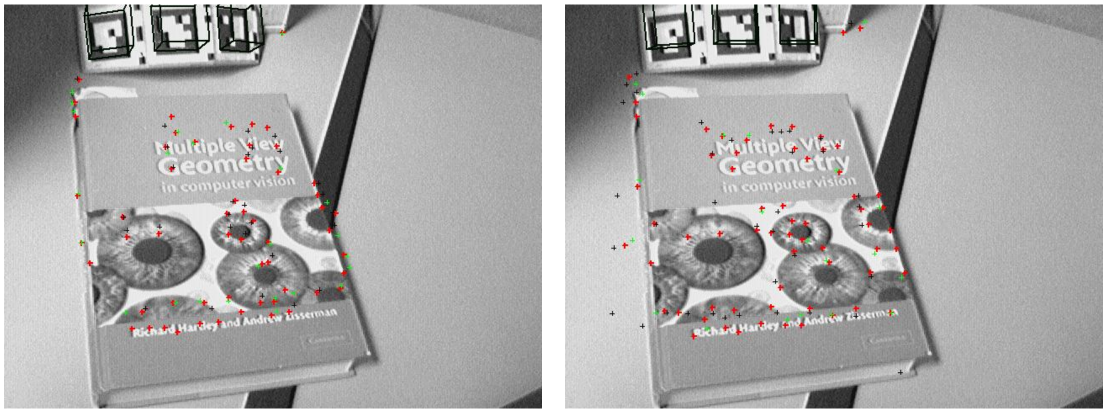

Multiple View Geometry in computer vision
Richard Hartley and Andrew Zisserman

Figure 5.7: Comparison of camera tracking at frame 130 for the planar scene. The fusion algorithm is used to get the pose and render the wireframe on the left and the HEIV algorithm is used for the right image.

the two estimates rapidly diverge. It is difficult to decide simply from this information which estimate is better. This is because we do not know the true world positions of any of the features now visible in the frames. Also, due to the fast camera motion, the gyroscope estimates are also fairly noisy and this makes the fusion estimates jittery. However, the fusion estimates are still less noisy than those returned by HEIV.

Around frame 350 the camera starts moving much slower and this makes the pose estimates much smoother and less jittery. Now, the extent of the drift suffered by the system is now gauged by bringing the markers back into view. The markers come back into view and are detected by the system at frame 520. At this time, the tracking phase ends and the markers are again used for pose computation. Since the markers are easily detectable and their world position is known, these pose estimates are reliable. The huge jump in the HEIV estimates shows that there has been a fair amount of drift. Although the fusion also suffers from drift it is much less in magnitude. The pose computed from triangulated features is used to render a wireframe around the markers at frame 519. These frames are shown in Figure 5.6 for both the simple HEIV method and the fusion method. Although our system suffers from some drift the wireframe still seems to be correct visually. However, for the HEIV system the wireframe is clearly wrong.

Pose estimation is an ill posed problem, especially when there is not too much depth

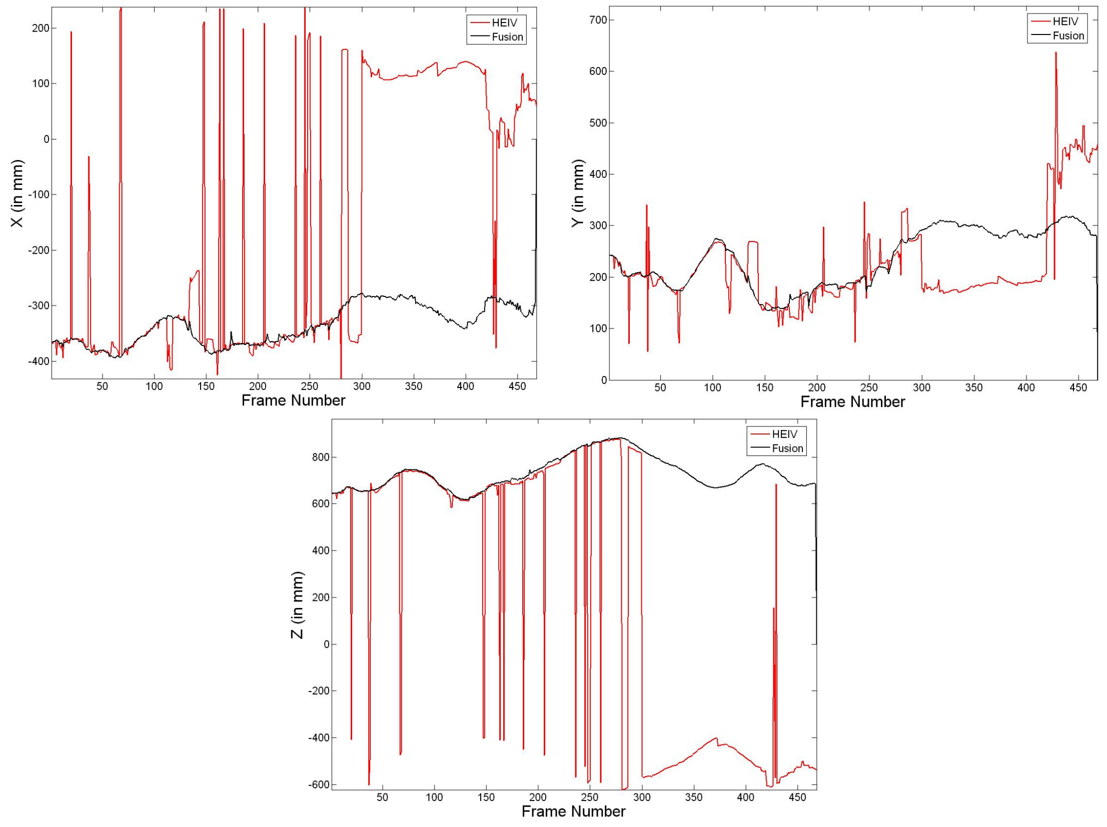


Figure 5.8: Camera paths results for experiment two. The simple HEIV algorithm breaks down and loses track due to the planarity of reconstructed features. The fusion algorithm presented has some jitter but is much more stable. The HEIV pose estimate keeps flipping between the true pose and its mirror image.

variation in the scene. Consequently, small errors in the orientation of the camera can be accounted for by small changes in the translational part of the pose. This implies that there are a number of pose estimates which can explain a given set of point correspondences. If the pose of each frame is estimated individually, we can get any one of these various pose estimates. It is possible to detect such a degeneracy, but doing this at every frame is computationally expensive. Modeling the camera dynamics offers a natural way to handle this problem. Even without a reparametrization, the availability of a rough orientation estimate prevents the pose from being completely incorrect. By enforcing the dynamics of the motion through the gyroscope, we make sure that the pose estimated for the current frame is not too different from the previous frame. This is a reasonable assumption since the camera moves in the real world and consequently

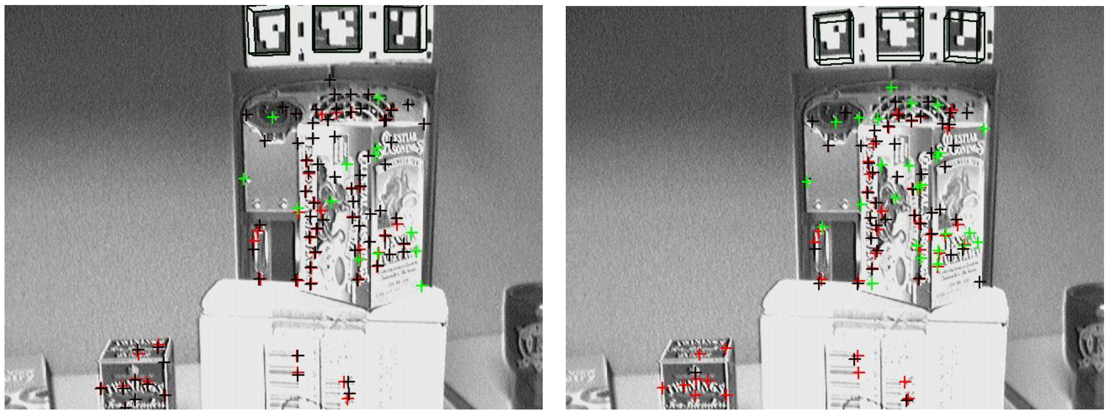


Figure 5.9: Comparison of camera tracking at frame 471 for experiment three. The fusion algorithm is used to get the pose and render the wireframe on the left and the HEIV algorithm is used on the right.

we get improved camera tracking results.

An example of a planar scene is shown in Figure 5.7. After a sufficient number of features have been triangulated satisfactorily, the feature pose is computed and used to render a wireframe around the markers. The results for one such frame are shown, using the simple HEIV method and the fusion method proposed here. The error of the HEIV method is clearly visible.

The planarity of reconstructed features frequently occurs in indoor scenes. Even if this degeneracy occurs only for a few frames, the pose estimates form those frames are used for the triangulation of further points. This leads to the errors multiplying and eventually causes drift. We consider such a scene in our next experiment where the sequence consists of 450 frames. Although multiple planar surfaces are present, it is possible that at times all the well triangulate features lie on a single surface which is planar or close to being planar. The camera tracks returned for this sequence by the two algorithms are compared in Figure 5.8. The HEIV method frequently loses track and suffer from large discontinuities. Note, how in the curve of the z-coordinate between frames 300 and 400 the two curves are mirror images of each other around the $z = 0$ line. Since the z-coordinate represents depth, this means that the HEIV estimate is returning the mirror image of the true pose due to the planarity of the scene. Enforcing the continuity of the motion prevents the fusion based system from jumping between

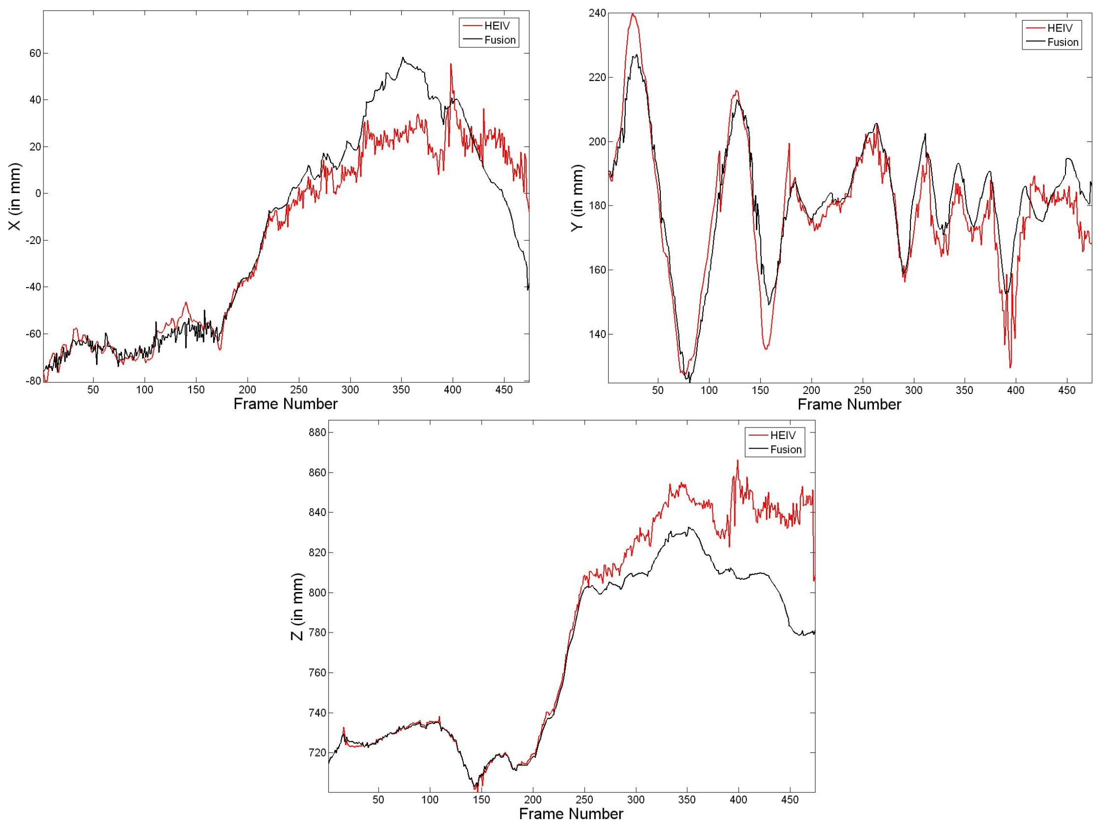


Figure 5.10: Camera paths results for experiment three. The fusion algorithm presented here clearly has a much more stable path and suffers from less drift.

the true pose and its mirror image.

Another advantage of enforcing the dynamics to get smoother camera path is that it allows us to handle faster camera motions as we show in the *third experiment*. On sequences where the simple HEIV method gives highly jittery paths and errors in reconstruction due to the speed of the camera, the fusion method is accurate and returns smooth camera paths. The results on an example of such camera motion are shown in Figures 5.9 and 5.10. The sequence consists of 474 frames. To get an idea of the drift graphically, the feature pose is used to render a wireframe around the markers in the frame before they reappear completely in the scene. These frames are shown in Figure 5.9. There is a considerable amount of drift in the simple HEIV based camera tracking and the wireframe is quite far from the markers. The fusion method does not suffer any noticeable amount of drift. The camera paths returned by the two systems are compared in Figure 5.10. The camera path returned by simple HEIV are clearly highly jittery.

The results shown here should not be taken as being representative of the HEIV method of [115]. Here we have specifically considered cases where it breaks down and our fusion algorithm still works. Although the fusion algorithm consistently does better than HEIV, the simple HEIV algorithm gives satisfactory results under fairly general conditions.

Chapter 6

Conclusion and Future Work

We have shown that the nonlinear constraints in vision problems lead to data which do not lie in vector spaces but on curved surfaces embedded in higher dimensional Euclidean spaces. These surfaces exhibit significant smoothness and can be studied using the theory of differential geometry. Using the appropriate methods which account for these nonlinearities can lead to advantages with theoretical consistency and improvements in practical performance of vision systems.

In this thesis, we proposed two new algorithms which account for the manifold nature of visual data. The nonlinear mean shift algorithm can be used for the clustering of data points lying on Riemannian manifolds. It was proved that the nonlinear mean shift iterations converge to a local maxima of a kernel density function over the manifold. Applications of this for motion segmentation and for discontinuity preserving filtering of complex image data were shown. The motion segmentation algorithm based on mean shift is a major improvement over previous work. To our knowledge, no previous algorithm has tried to solve the motion segmentation problem in such a general setting. The method of Section 3.6.1 does not require any knowledge of the number of motions present and can handle multiple motions and outliers. It estimates the number of motions present and the motion parameters in a single step. We have also obtained promising results when using mean shift for the filtering of complex image data. With the development of new imaging systems, especially for medical applications, image data will increase in complexity. Accounting for the nonlinear nature of the data will become necessary and we believe that the nonlinear mean shift algorithm will become as widely applied as the original mean shift algorithm.

The projection based M-estimator is a user independent regression algorithm which can adapt itself to varying levels of noise in the data set. It is able to handle both structured and unstructured outliers. With the increasing use of autonomous vision systems such user independence is necessary as the scene viewed by the system keep changing. There is an increasing amount of work being done to address the problem of making robust regression insensitive to user defined parameters and pbM is an important step in this direction. Previous modifications in robust regression have mainly concentrated on using external information to upgrade the sampling. Combining these methods with pbM, and the new robust cost function proposed in Section 4.2.3, should further improve results.

6.1 Future Work

Ideas for future work based on this thesis are discussed next.

6.1.1 The Radon Transform

As mentioned in Section 4.6, nonlinear mean shift and pbM can be used to solve the same problems but offer different advantages. Developing a robust estimator which can simultaneously estimate all the motions present in a data set without user intervention is still an open problem.

The major bottleneck of the mean shift technique is the hypothesis generation part of the algorithm [130]. We assume that if all the points used to generate a hypotheses are inliers then the hypothesis will be close to the true model. We refer to such hypotheses as *inlier hypotheses*. Let the fraction of inlier data points be $p < 1$, and let k points be required to generate a hypotheses. Then the fraction of inlier hypotheses will be p^k. Usually, we use some form of validation [20, 130], to improve the fraction of inlier hypotheses. In most data sets, especially those with multiple structures, p can be quite low and the algorithm does not scale very well with k. This occurs quite frequently in vision problems. For example, in fundamental matrix estimation each elementary subset consists of eight points ($k = 8$) and the fraction of inlier hypotheses can be quite small.

Although, mean shift does not scale to these problems, pbM still gives good results under such conditions. The pbM algorithm optimizes a cost function which is written in terms of the data points themselves rather than a set of hypotheses. Consequently, pbM requires a single hypotheses to be close enough to the true estimate rather then requiring a sizeable fraction of the hypotheses to be inliers.

The advantages of both methods can be obtained by proceeding like in pbM to generate a single hypotheses and then use nonlinear mean shift as a local optimization procedure for a cost function written in terms of the data points, *not in terms of other hypotheses*. Such a parameter estimation procedure is commonly known as the Radon transform [58]. The reason it is not widely used is that the maximization of the score in the parameter space is difficult.

We can use nonlinear mean shift for this maximization with some modifications. If $m(> 1)$ data points are required to uniquely define the model parameters, this implies that a single data point does not completely specify the model. Each data point defines a set of valid model parameters. For each correspondence, mean shift improves the current parameter estimate by finding the closest of all these parameters and moving towards it. This requires a characterization of the complete set of valid parameters based on a single data point. This characterization depends both on the manifold containing the parameters and the constraint relating the parameters and data. Also, given a set of valid parameters we need to find the closest point to our current estimate.

This is possible for simple manifolds such as the special Euclidean group for estimating 3D rigid motion or subspace estimation over the Grassmann manifolds, but extending this method to a larger class of applications is still an open problem.

6.1.2 Other Manifolds in Vision

Most manifolds we have considered here are fairly simple, low dimensional manifolds which have been also analyzed in other fields. However, there exist many different manifolds which are specific to computer vision but not very well understood. The essential manifold of Section 2.4.4 was an example of this. Another well known example of a manifold in computer vision is *shape space*, the space of all 2D curves invariant to rigid transformations [30] or to elastic deformations [53]. Although the theory has been known for a long time it is only recently that computers have developed enough to allow for an automated analysis based on these ideas. The theory of shape space is now being extended to 3D shape analysis [38].

Computer vision problems also lead to a number of less analyzed manifolds. Dimensionality reduction techniques such as locally linear embedding and Laplacian eigenmaps have shown that the space of visual images, in many cases, is much lower dimensional than was previously thought. These methods have also shown that this data does not follow the usual rules of Euclidean geometry. Analysis of cyclic processes such as human gait and heartbeats also form manifolds whose geometry is not very well understood, *e.g.* [33, 34].

A more principled analysis of these manifolds and their geometry is necessary. Once their geometry is well understood we can extend the methods developed here to these new manifolds for feature space analysis of new types of data.

Appendix A

Conjugate Gradient over $\mathbf{G}_{N,k} \times \mathbb{R}^k$

Most function optimization techniques, e.g., Newton iterations and conjugate gradient, apply to functions defined over Euclidean spaces. Similar methods have been developed for Grassmann manifolds [31]. As discussed in Section 4.4.2, the search space under consideration is the direct product of a Grassmann manifold and a real space, $\mathbf{G}_{N,k} \times \mathbb{R}^k$, and we want to perform conjugate gradient function minimization over this parameter space. The algorithm follows the same general structure as standard conjugate gradient but has some differences with regard to the movement of tangent vectors.

Let f be a real valued function on the manifold $\mathbf{G}_{N,k} \times \mathbb{R}^k$. Conjugate gradient minimization requires the computation of $\mathbf{G}$ and $\mathbf{g}$, the gradients of f with respect to $\boldsymbol{\Theta}$ and $\boldsymbol{\alpha}$. To obtain the gradients at a point $(\boldsymbol{\Theta}, \boldsymbol{\alpha})$, we compute the Jacobians $\mathbf{J}_{\boldsymbol{\Theta}}$ and $\mathbf{J}_{\boldsymbol{\alpha}}$ of f with respect to $\boldsymbol{\Theta}$ and $\boldsymbol{\alpha}$. The gradients are

$$\mathbf{G} = \mathbf{J}_{\boldsymbol{\Theta}} - \boldsymbol{\Theta}\boldsymbol{\Theta}^T\mathbf{J}_{\boldsymbol{\Theta}} \qquad\qquad \mathbf{g} = \mathbf{J}_{\boldsymbol{\alpha}}. \tag{A.1}$$

Let $[\boldsymbol{\Theta}_0, \boldsymbol{\alpha}_0] \in \mathbf{G}_{N,k} \times \mathbb{R}^k$ be the point at which the algorithm is initialized. Compute the gradients $\mathbf{G}_0$ and $\mathbf{g}_0$, at $(\boldsymbol{\Theta}_0, \boldsymbol{\alpha}_0)$ and the search directions are $\mathbf{H}_0 = -\mathbf{G}_0$ and $\mathbf{h}_0 = -\mathbf{g}_0$.

The following iterations are done till convergence. Iteration $j+1$ now proceeds by minimizing f along the geodesic defined by the search directions $\mathbf{H}_j$ on the Grassmann manifold and $\mathbf{h}_j$ in the Euclidean component of the parameter space. This is known as *line search*. The parametric form of the geodesic is

$$\boldsymbol{\Theta}_j(t) = \boldsymbol{\Theta}_j\mathbf{V}\mathbf{diag}(\cos\lambda t)\mathbf{V}^T + \mathbf{U}\mathbf{diag}(\sin\lambda t)\mathbf{V}^T \tag{A.2}$$

$$\boldsymbol{\alpha}_j(t) = \boldsymbol{\alpha}_j + t\mathbf{h}_j. \tag{A.3}$$

where, t is the parameter, $\boldsymbol{\Theta}_j$ is the estimate from iteration j and $\mathbf{U}\mathbf{diag}(\lambda)\mathbf{V}^T$ is the compact SVD of $\mathbf{H}_j$ consisting of the k largest singular values and corresponding singular vectors. The *sin* and *cos* act element-by-element [31].

Denoting the value of the parameter t where the minimum is achieved by t_m, set $\mathbf{\Theta}_{j+1} = \mathbf{\Theta}_j(t_m)$ and $\boldsymbol{\alpha}_{j+1} = \boldsymbol{\alpha}_j(t_m)$. The gradient vectors are parallel transported to this point according to

$$\mathbf{H}_j^\tau = [-\mathbf{\Theta}_j \mathbf{V}\mathbf{diag}(\sin\lambda t_m) + \mathbf{U}\mathbf{diag}(\cos\lambda t_m)]\mathbf{diag}(\lambda)\mathbf{V}^T$$
$$\mathbf{G}_j^\tau = \mathbf{G}_j - [\mathbf{\Theta}_j \mathbf{V}\mathbf{diag}(\sin\lambda t_m) + \mathbf{U}(\mathbf{I} - \mathbf{diag}(\cos\lambda t_m))]\mathbf{U}^T\mathbf{G}_j$$

The parallel transportation operator is denoted by τ. No explicit parallel transport is required for the Euclidean component of the parameter space since this is trivially achieved by moving the whole vector as it is. The new gradients $\mathbf{G}_{j+1}$ and $\mathbf{g}_{j+1}$ are computed at $(\mathbf{\Theta}_{j+1}, \boldsymbol{\alpha}_{j+1})$. The new search directions are chosen orthogonal to *all* previous search directions as

$$\mathbf{H}_{j+1} = -\mathbf{G}_{j+1} + \gamma_j \mathbf{H}_j^\tau \tag{A.4}$$

$$\mathbf{h}_{j+1} = -\mathbf{g}_{j+1} + \gamma_j \mathbf{h}_j \tag{A.5}$$

where

$$\gamma_j = \frac{tr\left(\left(\mathbf{G}_{j+1} - \mathbf{G}_j^\tau\right)^T \mathbf{G}_{j+1}\right) + (\mathbf{g}_{j+1} - \mathbf{g}_j)^T \mathbf{g}_{j+1}}{tr\left(\mathbf{G}_j^T \mathbf{G}_j\right) + \mathbf{g}_j^T \mathbf{g}_j} \tag{A.6}$$

and *tr* is the trace operator which computes the inner product between tangents of the Grassmann manifold. This value is unchanged by parallel translation and in the denominator we need not use the parallel transported tangents [31].

References

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, “Riemannian geometry of Grassmann manifolds with a view on algorithmic computation,” *Acta Applicandae Mathematicae*, vol. 80, no. 2, pp. 199–220, 2003.

[2] S. Amari and H. Nagaoka, *Methods of Information Geometry.* American Mathematical Society, 1st edition, 2001.

[3] L. P. Ammann, “Robust singular value decompositions: A new approach to projection pursuit,” *J. of Amer. Stat. Assoc.*, vol. 88, no. 422, pp. 505–514, 1993.

[4] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache., “Geometric means in a novel vector space structure on symmetric positive-definite matrices,” *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 1, pp. 328–347, 2006.

[5] A. Azarbayejani and A. Pentland, “Recursive estimation of motion, structure and focal length,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, no. 6, pp. 562–575, 1995.

[6] P. J. Basser, J. Mattiello, and D. LeBihan, “MR diffusion tensor spectroscopy and imaging,” *Biophysical Journal*, vol. 66, pp. 259–267, 1994.

[7] E. Begelfor and M. Werman, “Affine invariance revisited,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, vol. II, 2006, pp. 2087–2094.

[8] S. Birchfield and S. Rangarajan, “Spatiograms vs histograms for region-based tracking,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, vol. II, June 2005, pp. 1158–1163.

[9] W. M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry.* Academic Press, 2002.

[10] T. Broida and R. Chellappa, “Estimation of object motion parameters from noisy images,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 1, pp. 90–99, 1986.

[11] A. Buchanan and A. Fitzgibbon, “Damped newton algorithms for matrix factorization with missing data,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, vol. II, 2006, pp. 316–322.

[12] N. A. Campbell, “Robust procedures in multivariate analysis I: Robust covariance estimation,” *Applied Statistics*, vol. 29, no. 3, pp. 231–237, 1980.

[13] M. A. Carreira-Perpinan, “Gaussian mean-shift is an EM algorithm,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 5, pp. 767–776, 2007.

[14] H. Chen and P. Meer, “Robust regression with projection based M-estimators,” in *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France, vol. II, Oct 2003, pp. 878–885.

[15] H. Chen and P. Meer, “Robust fusion of uncertain information,” *IEEE Trans. Systems, Man, Cybernetics-Part B*, vol. 35, pp. 578–586, 2005.

[16] Y. Cheng, “Mean shift, mode seeking, and clustering,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 790–799, 1995.

[17] Y. Chikuse, *Statistics on Special Manifolds.* Springer, 2003.

[18] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, “Motion and structure causally integrated over time,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 4, pp. 523–535, 2002.

[19] C. M. Christoudias, B. Georgescu, and P. Meer, "Synergism in low level vision," in *Proc. 16th Intl. Conf. on Pattern Recognition,* Quebec, Canada, vol. IV, 2002, pp. 150–155.

[20] O. Chum and J. Matas, "Randomized RANSAC with $t_{d,d}$ test," in *British Machine Vision Conference*, 2002, pp. 448–457.

[21] O. Chum and J. Matas, "Matching with PROSAC - progressive sample consensus," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, vol. 1, June 2005, pp. 220–226.

[22] O. Chum, J. Matas, and J. Kittler, "Locally optimized RANSAC," in *DAGM Symposium Symposium for Pattern Recognition*, 2003, pp. 236–243.

[23] R. Collins, "Mean shift blob tracking through scale space," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI, vol. II, 2003, pp. 234–240.

[24] D. Comaniciu, "Variable bandwidth density-based fusion," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI, vol. 1, June 2003, pp. 59–66.

[25] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 603–619, May 2002.

[26] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, pp. 564–577, 2003.

[27] J. Costeira and T. Kanade, "A multi-body factorization method for motion analysis," in *Proc. 5th Intl. Conf. on Computer Vision,*Cambridge, MA, 1995, pp. 1071–1076.

[28] B. Davis, P. T. Fletcher, E. Bullitt, and S. Joshi, "Population shape regression from random design data," in *Proc. 11th Intl. Conf. on Computer Vision,* Rio de Janeiro, Brazil, Oct 2007.

[29] A. Davison, W. Mayol, and D. Murray, "Real-time localisation and mapping with wearable active vision," in *Proc. IEEE International Symposium on Mixed and Augmented Reality*, IEEE Computer Society Press, Oct. 2003, pp. 315–316.

[30] I. L. Dryden and K. V. Mardia, *Statistical Shape Analysis.* Wiley, 1st edition, 1998.

[31] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 2, pp. 303–353, 1998.

[32] A. Elgammal, R. Duraiswami, and L. S. Davis, "Efficient kernel density estimation using the efficient kernel density estimation using the color modeling and tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 11, pp. 1499–1504, 2003.

[33] A. Elgammal and C.-S. Lee, "Inferring 3D body pose from silhouettes using activity manifold learning," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, vol. II, 2004, pp. 681–688.

[34] A. Elgammal and C.-S. Lee, "Modeling view and posture manifolds for tracking," in *Proc. 11th Intl. Conf. on Computer Vision,* Rio de Janeiro, Brazil, 2007, pp. 1–8.

[35] M. Fashing and C. Tomasi, "Mean shift is a bound optimization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 3, pp. 471–474, 2005.

[36] R. Ferreira and J. Xavier, "Hessian of the Riemannian squared-distance function on connected locally symmetric spaces with applications," in *Controlo 2006, 7th Portuguese Conference on Automatic Control*, 2006.

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

[38] P. Fletcher, S. Joshi, C. Lu, and S. Pizer, "Principal geodesic analysis for the study of nonlinear statistics of shape," *IEEE Transactions on Medical Imaging*, vol. 23, no. 8, pp. 995–1005, 2004.

[39] P. T. Fletcher, C. Lu, and S. Joshi, "Statistics of shape via principal geodesic analysis on Lie groups," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI, 2003, pp. 95–101.

[40] W. Forstner and B. Moonen, "A metric for covariance matrices," Technical report, Dept. of Geodesy and Geoinformatices, Stuttgart University, 1999.

[41] J. M. Frahm and M. Pollefeys, "RANSAC for (quasi-)degenerate data (QDEGSAC)," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, vol. I, 2006, pp. 453–460.

[42] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* Academic Press, second edition, 1990.

[43] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Information Theory*, vol. 21, pp. 32–40, 1975.

[44] C. W. Gear, "Multibody grouping from motion images," *International J. of Computer Vision*, vol. 29, no. 2, pp. 133–150, 1998.

[45] A. Gelb, *Applied Optimal Estimation.* The M.I.T. Press, 8th edition, 1982.

[46] Y. Genc, S. Riedel, F. Souvannavong, C. Akinlar, and N. Navab, "Marker-less tracking for AR: A learning-based approach," in *IEEE / ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society, 2002, pp. 295–304.

[47] B. Georgescu and P. Meer, "Point matching under large image deformations and illumination changes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, pp. 674–689, 2004.

[48] C. Geyer, R. Bajcsy, and S. Sastry, "Euclid meets Fourier: Applying harmonic analysis to essential matrix estimation in omnidirectional cameras," in *Proc.of Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, 2004.

[49] G. H. Golub and C. F. Van Loan, *Matrix Computations.* The John Hopkins University Press, third edition, 1996.

[50] L. Goshen and I. Shimshoni, "Balanced exploration and exploitation model search for efficient epipolar geometry estimations," in *Proc. European Conf. on Computer Vision,* Graz, Austria, vol. II, 2006, pp. 151–164.

[51] V. M. Govindu, "Lie-algebraic averaging for globally consistent motion estimation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, vol. I, 2004, pp. 684–691.

[52] A. Graham, *Kronecker Products and Matrix Calculus:with Applications.* Ellis Horwood series in mathematics and its applications, 1981.

[53] U. Grenander, *General Pattern Theory: A Mathematical Study of Regular Structures.* Oxford University Press, 1st edition, 1993.

[54] A. Gruber and Y. Weiss, "Multibody factorization with uncertainty and missing data using the EM algorithm," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, vol. I, 2004, pp. 707–714.

[55] G. Hager, M. Dewan, and C. Stewart, "Multiple kernel tracking with SSD," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, vol. I, 2004, pp. 790–797.

[56] R. Haralick, C. Lee, K. Ottenberg, and M. Nolle, "Analysis and solutions of the three point perspective pose estimation problem," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Maui, HA, 1991, pp. 592–598.

[57] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[58] S. Helgason, *The Radon Transform.* Birkhuser Boston, 2nd edition, 1999.

[59] P. A. Helm, R. L. Winslow, and E. McVeigh, "Center for cardiovascular bioinformatics and modeling," Technical report, Johns Hopkins University.

[60] W. A. Hoff, "Fusion of data from head-mounted and fixed sensors," in *Proc. of First IEEE International Workshop on Augmented Reality*, November 1998.

[61] W. A. Hoff and T. Vincent, "Analysis of head pose accuracy in augmented reality," *IEEE Trans. Visualization and Computer Graphics*, vol. 6, no. 4, pp. 319–335, 2000.

[62] InterSense,Inc., *InterSense Wireless InertiaCube3*, 2006. Available at `http://www.intersense.com/products/prec/ic3/wirelessic3.htm`.

[63] I. T. Jolliffe, *Principal Component Analysis.* Springer, 2nd edition, 2002.

[64] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[65] K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice.* Elsevier, 1996.

[66] K. Kanatani, "Motion segmentation by subspace separation and model selection," in *Proc. 8th Intl. Conf. on Computer Vision,*Vancouver, Canada, vol. II, July 2001, pp. 301–306.

[67] Y. Kanazawa and K.Kanatani, "Do we really have to consider covariance matrices for image features?," in *Proc. 8th Intl. Conf. on Computer Vision,*Vancouver, Canada, vol. II, (Vancouver, CA), July 2001.

[68] H. Karcher, "Riemannian center of mass and mollifier smoothing," *Comm. Pure Appl. Math.*, vol. 30, no. 5, pp. 509–541, 1977.

[69] Q. Ke and T. Kanade, "Robust L-1 norm factorization in the presence of outliers and missing data by alternative convex programming," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, vol. I, 2005, pp. 739–746.

[70] D. LeBihan, J. F. Mangin, C. Poupon, C. A. Clark, S. Pappata, N. Molko, and H. Chabriat, "Diffusion tensor imaging: Concepts and applications," *Journal of Magnetic Resonance Imaging*, vol. 13, pp. 534–546, 2001.

[71] Y. Leedan and P. Meer, "Heteroscedastic regression in computer vision: Problems with bilinear constraint," *International J. of Computer Vision*, vol. 37, pp. 127–150, 2000.

[72] C. Lenglet, R. Deriche, and O. Faugeras, "Inferring white matter geometry from diffusion tensor MRI: Application to connectivity mapping," in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, vol. IV, 2004, pp. 127–140.

[73] R. Lenz and T. Bui, "Statistical properties of color signal spaces," *Journal of the Optical Society of America*, vol. 22, no. 5, pp. 820–827, 2005.

[74] R. Lenz, T. H. Bui, and J. Hernandez-Andres, "Group theoretical structure of spectral spaces," *Journal of Mathematical Imaging and Vision*, vol. 23, no. 3, pp. 297–313, 2005.

[75] R. Lenz and M. Solli, "Lie methods in color signal processing: Illumitnation effects," in *International Conference on Pattern Recognotion*, vol. III, 2006, pp. 738–741.

[76] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua, "Fully automated and stable registration for augmented reality applications," in *Proc. IEEE International Symposium on Mixed and Augmented Reality*, IEEE Computer Society Press, Nov. 2004, pp. 93–102.

[77] S. Li, "Robustizing robust M-estimation using deterministic annealing," *Pattern Recognition*, vol. 29, no. 1, pp. 159–166, 1996.

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

[79] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (DARPA)," in *Proceedings of the 1981 DARPA Image Understanding Workshop*, April 1981, pp. 121–130.

[80] Y. Ma, J. Kosecka, and S. Sastry, "Optimization criteria and geometric algorithms for motion and structure estimation," *International J. of Computer Vision*, vol. 44, no. 3, pp. 219–249, 2001.

[81] S. Mahamud, M. Herbert, Y. Omori, and J. Ponce, "Provably-convergent iterative methods for projective structure and motion," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, HI, vol. I, 2001, pp. 1018–1025.

[82] B. Matei, *Heteroscedastic Errors-In-Variables Models in Computer Vision.* PhD thesis, Department of Electrical and Computer Engineering, Rutgers University, 2001. Available at `http://www.caip.rutgers.edu/riul/research/theses.html`.

[83] B. Matei and P. Meer, "A general method for errors-in-variables problems in computer vision," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Hilton Head, SC, vol. II, June 2000, pp. 18–25.

[84] B. Matei and P. Meer, "Estimation of nonlinear errors-in-variables models for computer vision applications," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 28, no. 10, pp. 1537–1552, 2006.

[85] L. Matthies, R. Szelisky, and T. Kanade, "Kalman filter-based algorithms for estimating depth from image sequences," *International J. of Computer Vision*, vol. 3, no. 3, pp. 209–238, 1989.

[86] S. J. Maybank, *Theory of Reconstruction from Image Motion.* Springer-Verlag, 1992.

[87] P. Mordohai and G. Medioni, *Tensor voting: A Perceptual Organization Approach to Computer Vision and Machine Learning.* Morgan and Claypool Publishers, 2007.

[88] D. Nister, "Preemptive RANSAC for live structure and motion estimation," in *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France, vol. I, October 2003, pp. 199–206.

[89] D. Nister, "Preemptive RANSAC for live structure from motion," *Machine Vision and Applications*, vol. 16, no. 5, pp. 321–329, 2005.

[90] B. O'Neill, *Semi-Riemannian Manifolds:With Applications to Relativity.* Academic Press, 1983.

[91] B. Pelletier, "Kernel density estimation on Riemannian manifolds," *Statistics and Probability Letters*, vol. 73, no. 3, pp. 297–304, 2005.

[92] X. Pennec and N. Ayache, "Uniform distribution, distance and expectation problems for geomteric feature processing," *Journal of Mathematical Imaging and Vision*, vol. 9, no. 1, pp. 49–67, 1998.

[93] X. Pennec, P. Fillard, and N. Ayache, "A Riemannian framework for tensor computing," *International J. of Computer Vision*, vol. 66, no. 1, pp. 41–66, 2006.

[94] C. J. Poelman and T. Kanade, "A paraperspective factorization method for shape and motion recovery," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 3, pp. 206–218, 1997.

[95] M. Pollefeys, "Self calibration and metric reconstruction in spite of varying and unknown intrinsic camera parameters," *International J. of Computer Vision*, vol. 32, pp. 7–25, 1999.

[96] H. Qiu and E. R. Hancock, "Robust multi-body motion tracking using commute time clustering," in *Proc. European Conf. on Computer Vision,* Graz, Austria, vol. I, 2006, pp. 160–173.

[97] W. Rossmann, *Lie Groups: An Introduction through Linear Groups.* Oxford University Press, 2003.

[98] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection.* Wiley, 1987.

[99] S. Rozenfeld and I. Shimshoni, "The modified pbM-estimator method and a runtime analysis technique for the RANSAC family," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, vol. I, June 2005, pp. 1113–1120.

[100] P. D. Sampson, "Fitting conic sections to "Very Scattered" data: An iterative refinement of the Bookstein algorithm," *Computer Graphics and Image Processing*, vol. 18, pp. 97–108, 1982.

[101] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?"," in *Proceedings of the 7th European Conference on Computer Vision,* Copenhagen, Denmark, vol. 1, 2002, pp. 414–431.

[102] J. Selig, *Geometrical Methods in Robotics.* Springer-Verlag, 1st edition, 1996.

[103] G. Simon, A. Fitzgibbon, and A. Zisserman, "Markerless tracking using planar structures in the scene," in *Proc. International Symposium on Augmented Reality*, Oct. 2000, pp. 120–128.

[104] M. Singh and N. Ahuja, "Regression based bandwidth selection for segmentation using Parzen windows," in *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France, vol. I, October 2003, pp. 2–9.

[105] M. Singh, H. Arora, and N. Ahuja, "A robust probabilistic estimation framework for parametric image models," in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, vol. I, May 2004, pp. 508–522.

[106] S. T. Smith, "Optimization techniques on Riemannian manifolds," *Fields Institute Communications*, vol. 3, pp. 113–146, 1994.

[107] S. Soatto, P. Perona, R. Frezza, and G. Picci, "Motion estimation via dynamic vision," in *Proc. European Conf. on Computer Vision,* Stockholm, Sweden, vol. II, 1994, pp. 61–72.

[108] C. V. Stewart, "Robust parameter estimation in computer vision.," *SIAM Reviews*, vol. 41, pp. 513–537, 1999.

[109] R. Subbarao, Y. Genc, and P. Meer, "Robust unambiguous parametrization of the essential manifold," in *In preparation*, 2007.

[110] R. Subbarao and P. Meer, "Heteroscedastic projection based M-estimators," in *Workshop on Empirical Evaluation Methods in Computer Vision,* San Diego, CA, June 2005.

[111] R. Subbarao and P. Meer, "Beyond RANSAC: User independent robust regression," in *Workshop on 25 Years of RANSAC,* New York, NY, June 2006.

[112] R. Subbarao and P. Meer, "Nonlinear mean shift for clustering over analytic manifolds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, vol. I, 2006, pp. 1168–1175.

[113] R. Subbarao and P. Meer, "Subspace estimation using projection based M-estimators over Grassmann manifolds," in *Proc. European Conf. on Computer Vision,* Graz, Austria, vol. I, May 2006, pp. 301–312.

[114] R. Subbarao and P. Meer, “Discontinuity preserving filtering over analytic manifolds,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Minneapolis, MN, 2007.

[115] R. Subbarao, P. Meer, and Y. Genc, “A balanced approach to 3D tracking from image streams,” in *Proc. IEEE and ACM International Symposium on Mixed and Augmented Reality*, October 2005, pp. 70–78.

[116] Y. Sugaya and K. Kanatani, “Multi-stage unsupervised learning for multibody motion segmentaion,” in *Proceedings of 2003 Workshop on Information-Based Induction Sciences (IBIS 2003)*, 2003, pp. 113–118.

[117] Y. Sugaya and K. Kanatani, “Geometric structure of degeneracy for multi-body motion segmentation,” in *The 2nd Workshop on Statistical Methods in Video Processing (SMVP 2004)*, number 3247 in LNCS, pp. 13–25, Springer–Verlag, December 2004.

[118] C. Tomasi and T. Kanade, “Shape and motion from image streams under orthography: A factorization method,” *International J. of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[119] C. Tomasi and J. Shi, “Good features to track,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR’94)*, (Seattle, WA), 1994, pp. 593–600.

[120] B. Tordoff and D. Murray, “Guided sampling and consensus for motion estimation,” in *Proc. European Conf. on Computer Vision,* Copenhagen, Denmark, vol. I, May 2002, pp. 82–96.

[121] P. H. S. Torr and C. Davidson, “IMPSAC: Synthesis of importance sampling and random sample consensus,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 3, pp. 354–364, 2003.

[122] P. H. S. Torr and D. W. Murray, “The development and comparison of robust methods for estimating the fundamental matrix,” *International J. of Computer Vision*, vol. 24, no. 3, pp. 271–300, 1997.

[123] P. H. S. Torr and A. Zisserman, “Robust parameterization and computation of the trifocal tensor,” *Image and Vision Computing*, vol. 15, pp. 591–605, August 1997.

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

[125] B. Triggs, “Factorization methods for projective structure and motion,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Francisco, CA, vol. I, 1996, pp. 845–851.

[126] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, “Bundle adjustment — A modern synthesis,” in B. Triggs, A. Zisserman, and R. Szelisky, editors, *Vision Algorithms: Theory and Practice*, Springer, 2000, pp. 298–372.

[127] R. Tron and R. Vidal, “A benchmark for the comparison of 3-d motion segmentation algorithms,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Minneapolis, MN, 2007.

[128] R. Tsai, “A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses,” *IEEE Journal of Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[129] D. Tschumperle and R. Deriche, “Vector-valued image regularization with PDEs: A common framework for different applications,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 4, pp. 506–517, 2005.

[130] O. Tuzel, R. Subbarao, and P. Meer, “Simultaneous multiple 3D motion estimation via mode finding on Lie groups,” in *Proc. 10th Intl. Conf. on Computer Vision,* Beijing, China, vol. 1, 2005, pp. 18–25.

[131] S. Van Huffel and J. Vanderwalle, “Analysis and properties of GTLS in problem $AX \approx B$,” *SIAM Journal on Matrix Analysis and Applications*, vol. 10, pp. 294–315, 1989.

[132] B. C. Vemuri, Y. Chen, M. Rao, T. McGraw, Z. Wang, and T. Mareci, “Fiber tract mapping from diffusion tensor MRI,” in *In Proceedings of the IEEE Workshop on Variational and Level Set Methods*, 2001, pp. 81–88.

[133] R. Vidal and Y. Ma, “A unified algebraic approach to 2-D and 3-D motion segmentation,” in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, vol. I, 2004, pp. 1–15.

[134] R. Vidal, Y. Ma, and J. Piazzi, “A new GPCA algorithm for clustering subspaces by fitting, differentiating and dividing polynomials,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, vol. I, 2004, pp. 510–517.

[135] R. Vidal, Y. Ma, and S. Sastry, “Generalized principal component analysis (GPCA).,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 12, pp. 1–15, 2005.

[136] M. P. Wand and M. C. Jones, *Kernel Smoothing.* Chapman and Hall, 1995.

[137] H. Wang and D. Suter, “Robust adaptive-scale parametric model estimation for computer vision,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 11, pp. 1459–1474, 2004.

[138] H. Wang and D. Suter, “Robust fitting by adaptive-scale residual consensus,” in T. Pajdla and J. Matas, editors, *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, vol. III, Springer-Verlag, May 2004, pp. 107–118.

[139] J. Wang, B. Thiesson, Y. Xu, and M. Cohen, “Image and video segmentation by anisotropic kernel mean shift,” in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, vol. II, 2004, pp. 238–249.

[140] C. Yang, R. Duraiswami, D. DeMenthon, and L. Davis, “Mean-shift analysis using quasi-Newton methods,” in *Intl. Conf. on Image Processing*, vol. II, 2003, pp. 447–450.

[141] L. Zelnik-Manor and M. Irani, “Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI, vol. 2, 2003, pp. 287–293.

[142] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, 2000.

[143] Z. Zivkovic and B. Krose, “An EM-like algorithm for color-histogram-based object tracking,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC, vol. I, June 2004, pp. 798–803.

Vita

Raghav Subbarao

Education

2003-2008	Ph.D., Department of Electrical and Computer Engineering, Rutgers University
1999-2003	Bachelor of Technology, Department of Electrical Engineering, Indian Institute of Technology, New Delhi

Publications

- **R. Subbarao** and P. Meer, "Discontinuity preserving filtering over analytic manifolds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Minneapolis, MN, 2007.
- **R. Subbarao**, Y. Genc, and P. Meer, "Nonlinear mean shift for robust pose estimation," in *8th IEEE Workshop on Applications of Computer Vision,* Austin, TX, February 2007.
- **R. Subbarao** and P. Meer, "Beyond RANSAC: User independent robust regression," in *Workshop on 25 Years of RANSAC,* New York, NY, June 2006.
- **R. Subbarao** and P. Meer, "Nonlinear mean shift for clustering over analytic manifolds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY, vol. I, 2006, pp. 1168–1175.
- **R. Subbarao** and P. Meer, "Subspace estimation using projection based M-estimators over Grassmann manifolds," in *Proc. European Conf. on Computer Vision,* Graz, Austria, vol. I, May 2006, pp. 301–312.
- S. Kalyan Krishna, **R. Subbarao**, S. Chaudhury and A. Kumar, Parsing News Video Using Integrated Audio-Video Features, *1st International Conference on Pattern Recognition and Machine Intelligence*, Calcutta, India, December 2005, 538-543.
- O. Tuzel, **R. Subbarao**, and P. Meer, "Simultaneous multiple 3D motion estimation via mode finding on Lie groups," in *Proc. 10th Intl. Conf. on Computer Vision,* Beijing, China, vol. 1, 2005, pp. 18-25.

- **R. Subbarao**, P. Meer, and Y. Genc, "A balanced approach to 3D tracking from image streams," in *Proc. IEEE and ACM International Symposium on Mixed and Augmented Reality*, October 2005, pp. 70-78.
- **R. Subbarao** and P. Meer, "Heteroscedastic projection based M-estimators," in *Workshop on Empirical Evaluation Methods in Computer Vision,* San Diego, CA, June 2005.

Submitted to Journal

- **R. Subbarao** and P. Meer, Nonlinear Mean Shift Over Riemannian Manifolds. *submitted to International Journal of Computer Vision*, 2008.
- **R. Subbarao** and P. Meer, Projection Based M-Estimators. *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.