Multiple Camera Management Using Wide Baseline Matching

by

Ruan Lakemond, BEng (Hons, 1st Class)

PhD Thesis

Submitted in Fulfilment

of the Requirements

for the Degree of

Doctor of Philosophy

at the

Queensland University of Technology

Image and Video Research Laboratory

Faculty of Built Environment and Engineering

January 2010

Keywords

Image processing, computer vision, projective geometry, image registration, feature extraction, local image features, wide baseline matching, optical flow, sparse optical flow.

Abstract

Camera calibration information is required in order for multiple camera networks to deliver more than the sum of many single camera systems. Methods exist for manually calibrating cameras with high accuracy. Manually calibrating networks with many cameras is, however, time consuming, expensive and impractical for networks that undergo frequent change. For this reason, automatic calibration techniques have been vigorously researched in recent years.

Fully automatic calibration methods depend on the ability to automatically find point correspondences between overlapping views. In typical camera networks, cameras are placed far apart to maximise coverage. This is referred to as a wide base-line scenario. Finding sufficient correspondences for camera calibration in wide base-line scenarios presents a significant challenge.

This thesis focuses on developing more effective and efficient techniques for finding correspondences in uncalibrated, wide baseline, multiple-camera scenarios. The project consists of two major areas of work. The first is the development of more effective and efficient view covariant local feature extractors. The second area involves finding methods to extract scene information using the information contained in a limited set of matched affine features.

Several novel affine adaptation techniques for salient features have been developed. A method is presented for efficiently computing the discrete scale space primal sketch of local image features. A scale selection method was implemented that makes use of the primal sketch. The primal sketch-based scale selection method has several advantages over the existing methods. It allows greater freedom in how the scale space is sampled, enables more accurate scale selection, is more effective at combining different functions for spatial position and scale selection, and leads to greater computational efficiency.

Existing affine adaptation methods make use of the second moment matrix to estimate the local affine shape of local image features. In this thesis, it is shown that the Hessian matrix can be used in a similar way to estimate local feature shape. The Hessian matrix is effective for estimating the shape of blob-like structures, but is less effective for corner structures. It is simpler to compute than the second moment matrix, leading to a significant reduction in computational cost.

A wide baseline dense correspondence extraction system, called WiDense, is presented in this thesis. It allows the extraction of large numbers of additional accurate correspondences, given only a few initial putative correspondences. It consists of the following algorithms: An affine region alignment algorithm that ensures accurate alignment between matched features; A method for extracting more matches in the vicinity of a matched pair of affine features, using the alignment information contained in the match; An algorithm for extracting large numbers of highly accurate point correspondences from an aligned pair of feature regions. Experiments show that the correspondences generated by the WiDense system improves the success rate of computing the epipolar geometry of very widely separated views. This new method is successful in many cases where the features produced by the best wide baseline matching algorithms are insufficient for computing the scene geometry.

Contents

List of Tables

List of Figures

List of Algorithms

Notation

c	A scalar value.
$\mathbf{x}$	A column vector.
$\mathbf{M}_{m\times n}$	A matrix with m rows and n columns.
$\mathbf{M}$	A matrix with number of rows and columns previously defined.
$\mathbf{I}$	The identity matrix – a square matrix with the diagonal entries equal to one and all other entries equal to zero.
$\lvert\mathbf{M}\rvert$	The determinant of the matrix $\mathbf{M}$.
$\operatorname{trace}(\mathbf{M})$	The trace of matrix $\mathbf{M}$. The trace of a matrix is the sum of the diagonal elements.
$\mathbf{T}(t_x, t_y)$	A translation transformation.
$\mathbf{R}(\theta)$	A rotation transformation.
$\mathbf{K}(k)$	An isotropic scaling transformation.
$\mathbf{A}(q, \phi)$	An anisotropic scaling transformation.
$\mathbf{x}_1 \cdot \mathbf{x}_2 = \mathbf{x}_1^\top \mathbf{x}_2$	The inner product or scalar product of column vectors $\mathbf{x}_1$ and $\mathbf{x}_2$.
$\mathbf{x}_1 \times \mathbf{x}_2$	The vector product or cross product of column vectors $\mathbf{x}_1$ and $\mathbf{x}_2$.

$[\mathbf{x}]_\times$	The skew matrix corresponding to vector $\mathbf{x}$. Defined as, $$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix},$$ for a three vector $\mathbf{x} = [x_1, x_2, x_3]^\top$. Can be used to evaluate the vector cross product, $\mathbf{x}_1 \times \mathbf{x}_2 = [\mathbf{x}_1]_\times \mathbf{x}_2$.
$I(\mathbf{x})$	A function on the vector $\mathbf{x}$. Often used to represent images.
$I_1(\mathbf{x}) * I_2(\mathbf{x})$	The convolution of functions $I_1(\mathbf{x})$ and $I_2(\mathbf{x})$.

Acronyms & Abbreviations

2D	Two-dimensional
3D	Three-dimensional
DOLP	Difference of Low Pass
DoG	Difference of Gaussians
EBR	Edge Based Regions
GLOH	Gradient Location-Orientation Histogram
IBR	Intensity Based Regions
IIR	Infinite Impulse Response
IP	Internet Protocol
JPEG	An image compression standard named after the Joint Photographic Experts Group
LoG	Laplacian of Gaussians
LMS	Least Median of Squares
MSER	Maximally Stable Extremal Regions
RANSAC	Random Sample Consensus
RIFT	Rotation Invariant Feature Transform

RMS	Root Mean Squared
PCA	Principal Component Analysis
SAF	Stable Affine Frames
SIFT	Scale Invariant Feature Transform
SSFS	Scale Space Feature Sketch
SURF	Speed Up Robust Features
SVD	Singular Value Decomposition

Certification of Thesis

The work contained in this thesis has not been previously submitted for a degree or diploma at any other higher educational institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed:	______________________
Date:	______________________

Acknowledgments

I would like to thank my supervisors, Prof. Sridha Sridharan and Dr. Clinton Fookes, for their support and guidance throughout my PhD candidature, and for convincing me to undertake a PhD in the first place. They provided access to excellent facilities and a knowledgeable and helpful group of colleagues.

I would like to thank my wife for her dedicated and loving support. She has provided consistent joy, support, discipline, motivation and encouragement throughout the PhD process.

I would like to thank my family for their support throughout my education. They have made it possible for me to pursue a career path that is well suited to my interests, ambitions and aptitude.

Finally, I would like to thank my colleagues in the SAIVT group and express my gratitude for their cooperative culture. Each one contributed to a productive and enjoyable work environment and provided support, insights, ideas and motivation on the path to completing this project.

Ruan Lakemond

Queensland University of Technology
January 2010

Chapter 1

Introduction

A vast collection of computer vision research is focused on single camera implementations. This collection is progressively being extended to multiple camera networks. In the past, camera networks consisted of sparsely placed cameras with little or no overlap between camera views. The availability of inexpensive sensors and the declining price of IP cameras enable networks to be designed with significant overlap between views. Camera networks with overlapping views provide far greater possibilities for computer vision applications than single camera systems. Three-dimensional (3D) scene analysis is just one example.

In order for multiple camera networks to deliver more than the sum of many single camera systems, the relationships between cameras must be known and exploited. This requires camera calibration information. Methods exist to manually calibrate cameras with high accuracy. Manual calibration of networks with many cameras is time consuming, expensive and impractical for networks that undergo frequent change. For this reason, automatic calibration techniques have been vigorously researched in recent years.

Fully automatic calibration methods depend on the ability to automatically find point correspondences between overlapping views. Even in networks where large numbers of cameras are available, the cameras are typically placed far apart in the surveillance environment. The problem of finding sufficient correspondences for camera calibration in scenarios where cameras are sparsely placed, presents a significant challenge. This PhD project focuses on developing more effective and efficient techniques for finding correspondences in difficult, uncalibrated, wide base-line, multiple camera scenarios.

1.1 Motivation

Camera networks where multiple cameras view the same scene, or where the viewing area of the cameras partially overlap, provide far greater opportunities for creating useful computer vision systems than a similar network with no overlap. In a single view system, any form of scene analysis is strictly limited to two-dimensional (2D) analysis in an arbitrary or poorly registered coordinate system. In a multiple view system (where overlapping views are available), it is possible to reconstruct the scene in three dimensions, register the reconstruction to a real building model and accurately track elements in the scene in terms of the building coordinates, as they move through the building. Being able to reconstruct the scene in three dimensions and register it to real world coordinates, enables much more accurate measurements to be made. Example applications that can benefit from 3D data and overlapping views include object tracking, human pose reconstruction, crowd and queue monitoring, action recognition, person recognition (using biometrics such as gate and face) and event detection. In particular, many of the applications listed above suffer from problems related to subject pose, where only 2D data is available. The availability of multiple views and 3D data provide a greater ability to estimate the pose of subjects, deal with occlusions and extract the information required to perform a given task.

Very few of the benefits of overlapping views are available without camera calibration information. Camera calibration consists of the parameters of a model that describes the camera position, orientation and operation. Without this information there is no basis for transferring information between views. Methods exist for manually calibrating cameras with high accuracy. Manually calibrating networks with many cameras is, however, time consuming, expensive and impractical for networks that undergo frequent change. The prospect of automating the calibration process is very attractive and has inspired much research.

Fully automatic calibration methods depend on the ability to automatically find point correspondences between overlapping views. Where two cameras are placed close together and have similar focal lengths (short base-line), the two images are very similar and finding correspondences is a simple problem. In typical camera networks, cameras are placed far apart in order to cover a larger area. This is referred to as a wide base-line scenario. Finding sufficient correspondences for camera calibration in wide base-line scenarios presents a significant challenge.

Figure[1] 1.1 illustrates the wide baseline matching problem. In the small baseline case, each point in the left image can be matched with a corresponding point in the right image simply by comparing local image regions directly. The wide baseline case, on the other hand, involves significant appearance changes, self occlusion, changed relative order of objects and photometric variations. The third row of Figure 1.1 shows a real surveillance scenario. It is difficult, even for a human observer, to find correspondences between these views. The research presented in this thesis aims to address this wide baseline matching problem in order to enable automatic calibration.

[1]Images sourced from `http://www.cs.unc.edu/~marc/data/castlejpg.zip` and `http://www.cvg.rdg.ac.uk/PETS2006/data.html`.

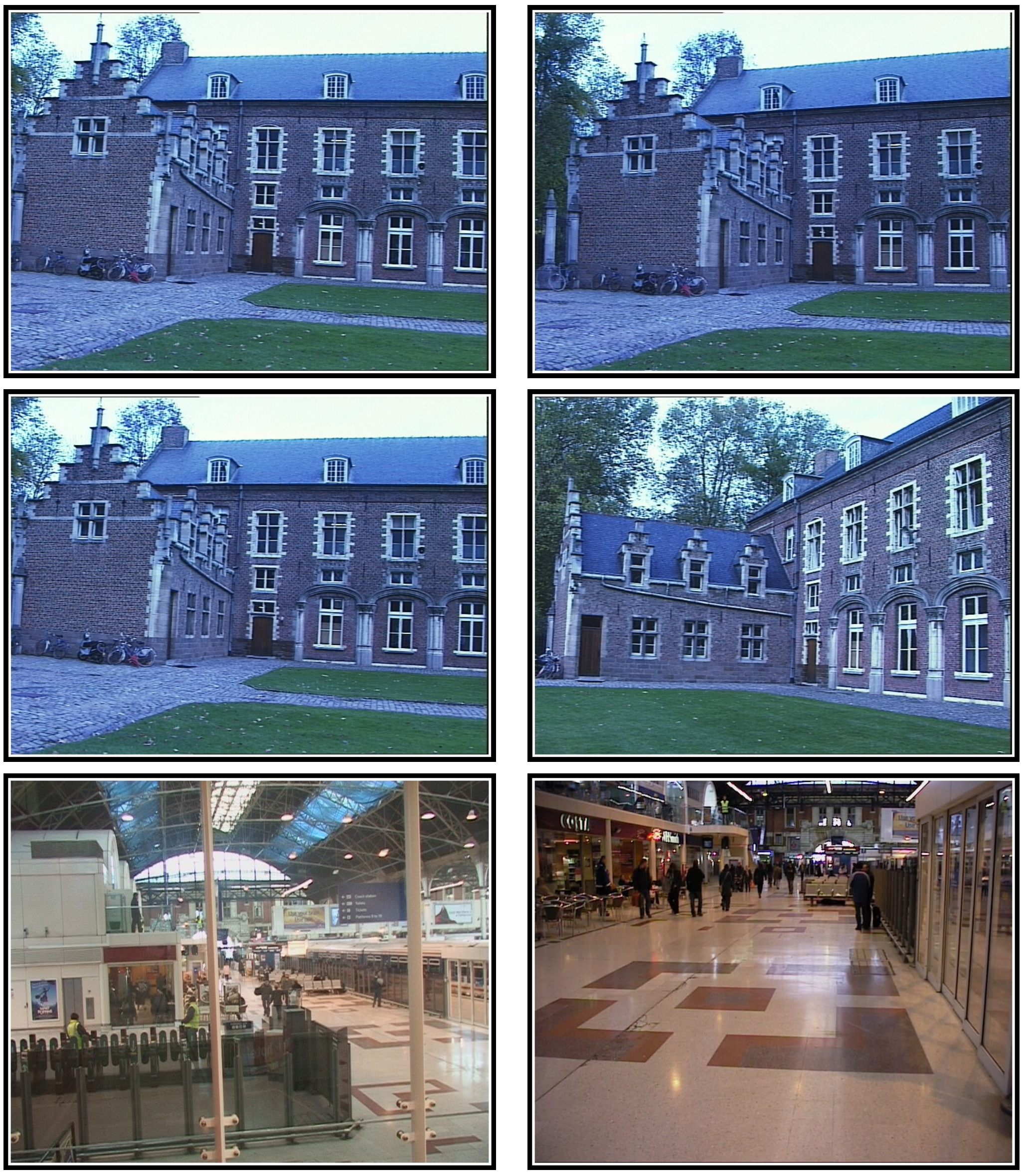


Figure 1.1: A small baseline image pair (top), wide baseline image pair (middle), and a real pair of surveillance images.

1.2 Aims and Objectives

1.2.1 Scope

Addressing the problem of automatic camera calibration requires solving a large array of sub-problems spanning many disciplines, including image processing, computer vision and projective geometry. This thesis is focussed on the problem of finding correspondences between multiple overlapping views. Finding correspondences is the first step towards automatic calibration.

The existing wide baseline matching methods may be divided into three sections: local feature extraction, featured description and descriptor matching. Feature extraction forms the first of two focus areas in this thesis. More specifically, feature extraction techniques are developed that use fewer processing resources and yield more robust features.

It is expected that after the field of feature-based matching reaches maturity, there will still be scenarios where feature-based matching is insufficient for computing the scene geometry. The second focus of this thesis is to explore new methods, beyond feature-based matching, for extracting correspondences from overlapping views. Methods are developed that make use of the information extracted using feature-based matching to commence a second tier of correspondence extraction.

In summary, this thesis focuses on two specific research tasks: Improving the quality and efficiency of local feature extractors; and exploring a second tier of correspondence extraction algorithms capable of exploiting the limited information discovered through feature matching.

1.2.2 Objectives

The primary research objectives for this thesis are to:

1. Develop local feature extraction techniques that improve on the robustness and computational efficiency of existing wide baseline methods.
2. Develop second tier correspondence extraction methods that exploit the information contained in a set of putative feature correspondences to further extract correspondences from multiple views.

The first objective is motivated by the need to deal with more challenging scenarios and by the fact that some of the best existing local feature extractors incur a relatively high computational cost. Local feature methods have steadily been improving over the last few decades. There are further gains to be made in terms of scale and affine covariance.

The second objective recognises that there will always be a limit to the ability of feature-based matching to solve complex registration problems. In order to improve correspondence extraction algorithms, it is necessary to investigate alternative methods that can be implemented in addition to feature-based matching. Affine covariant feature extractors produce a significant amount of information that is currently not utilised by geometry computation algorithms. Matched affine features contain information about the relative shape of features. Current geometry computation methods only use point correspondences. A new generation of methods are developed in this thesis to exploit all the information contained in matched affine features and to produce additional correspondences. These methods are referred to as second tier correspondence extraction methods – new methods that extract further information once the first tier of methods (existing feature-based matching) have been completed.

1.3 Outline of the Thesis

Chapter 2

This chapter introduces some of the basic concepts in projective geometry that are used throughout this thesis. Projective geometry is the study of geometric properties which are invariant under projective transformations. It encompasses camera geometry and the formation of images through 3D to 2D projection. The algebra of projective geometry serves not only to formally express the concepts of projective geometry, but is also of practical use in designing image processing algorithms.

The subject will be treated in a manner that provides the understanding and algebraic tools required to achieve the objective of finding correspondences between views. Geometric invariance properties will not be discussed in great detail. The focus will instead be on the practical implications for image processing.

The concepts covered include coordinate transformations and their application to image processing, the pinhole camera model, epipolar geometry and approaches to camera calibration.

Chapter 3

The problem of camera calibration may be interpreted as a problem of aligning two views of a scene by means of a projection model. It is also possible to align subregions of the images, where the relationship between these image regions is much simpler. Chapter 3 formulates the problem of aligning images taken from different viewpoints in terms of the wide baseline matching problem. Local image feature extraction and matching provides a solution to the wide baseline image alignment problem. Local image features are patterns in an image that are defined in limited image areas and are distinguishable from the surrounding image in some way. They may be extracted from different views independently and then matched to find corresponding features. The discussion in Chapter 3 focuses on feature-based matching in the wide baseline scenario, though feature-based methods are suitable for many applications. Feature extraction, description and matching techniques relevant to wide baseline matching are reviewed, as well as methods for evaluating the performance of local feature extractors.

Chapter 4

A number of successful affine covariant feature extractors make use of a saliency map to detect features, and use adaptation processes to extract the shape and size of each feature. Chapter 4 reviews saliency map-based feature extraction techniques.

The saliency map can be used to find feature locations, but does not directly produce features that are sufficiently robust for wide baseline matching. In order to make features more robust, a characteristic scale, shape and orientation must be found for each feature. The Gaussian scale space is introduced and the literature pertaining to scale, shape and orientation selection is reviewed. The work presented in Chapters 5 and 6 develops new methods for feature scale and shape selection. An algorithmic framework for feature extraction is presented in Chapter 4 that is used to compare the performance of the newly developed methods. Lastly, methods for implementing some of the scale space and saliency operators in an efficient manner are discussed.

Chapter 5

The concept of scale plays an important role in wide baseline vision. Characteristic scale selection is an effective method to deal with the scale problem in the domain of robust local feature extraction. Chapter 4 discusses methods proposed in the literature to select a characteristic scale for each local image feature. In Chapter 5, the scale space primal sketch is employed as a tool for analysing features in scale space.

The scale space primal sketch has in the past been a useful tool for multi-scale analysis of features, and for scale selection. A method is developed for computing a discrete representation of the scale space primal sketch of local image features. This discrete primal sketch is used to implement a more efficient and effective method for scale selection, by combining the sketch with modern feature extraction and scale selection techniques. The performance of this algorithm is compared to existing methods. It is shown that the new method poses several advantages over existing scale selection methods.

Chapter 6

The second moment matrix is the most successful shape estimator used in affine adaptation of saliency map-based local image features. Chapter 6 explores the novel approach of using the Hessian matrix as an affine shape measure for affine adaptation. The Hessian is simpler to implement and requires less computational effort than the second moment matrix. Experiments show that it is also more effective in practical problems in combination with a feature extractor that extracts blob-like regions. The main limitation of the Hessian matrix is that it is unsuitable for use with corner features.

The ability of the Hessian matrix to measure affine shape is demonstrated from two points of view. It is first shown how the Hessian matrix can be used to measure the covariance matrix of a Gaussian blob up to scale. Secondly, it is shown that it can be used to measure the shape of structures in scale space with arbitrary shape. In both cases the model is an approximation to the real image feature shape, and the process must be applied iteratively in order to find the true affine shape of a feature.

The complexity and performance of the Hessian matrix and second moment matrix are compared. The Hessian matrix requires significantly less processing time than the second moment matrix. When used in combination with a blob-like feature extractor, the Hessian produces equivalent quality features to the second moment matrix; however, when used in combination with a corner extractor, the feature quality is significantly reduced.

Chapter 7

The affine covariant feature extractors investigated and developed in earlier chapters are robust and effective in extracting correspondences between wide baseline views. Despite the quality of modern correspondence extraction methods, they can not provide sufficient correspondences in all camera and scene configurations. If cameras are too sparsely placed, produce low resolution images, or if the scene contains few distinguishable features, then wide baseline matching techniques may not be able to produce sufficient correspondences to allow computation of the camera geometry.

Chapter 7 presents the Uncalibrated Wide Baseline Dense Optical Flow algorithm, called WiDense. It is a method for extracting large numbers of accurate correspondences between a pair of views using only a set of putative wide baseline correspondences as input. This new method does not rely on any knowledge of the camera geometry, only the information contained in matches produced through wide baseline matching. It can produce a useful set of correspondences even when the input set of affine matches is not sufficient to compute the epipolar geometry.

Chapter 7 also presents a method for evaluating wide baseline correspondence extraction systems. The evaluation system attempts to compute the epipolar geometry of a set of scenes using the output of a given wide baseline matching system. The results are presented in terms of average proportion of successful trials. This evaluation method is used to compare the performance of the WiDense algorithm to existing wide baseline matching techniques. It is shown through this evaluation that the correspondences generated by the WiDense algorithm can be used to compute the epipolar geometry in many cases where existing wide baseline methods alone do not provide sufficient reliable correspondences.

Chapter 8

The final chapter presents the conclusions of the thesis, and considers possible future directions.

1.4 Original Contributions of the Thesis

Preliminaries

- A review of feature extraction and matching techniques is presented. The review focuses specifically on affine covariant features suitable for wide baseline matching.

- A common algorithmic framework for saliency-based feature extraction and adaptation is presented. This framework is used to evaluate novel methods presented in the thesis. It allows individual modules to be exchanged easily, so that extraction and adaptation techniques can be evaluated on a common platform.

Scale Selection

A novel scale selection method is presented in this thesis that combines the scale space primal sketch and modern scale selection methods. To this end, the following contributions are made:

- The existing scale space primal sketch concept is modified to make use of saliency map-based features as primitives, creating the scale space feature sketch.
- An efficient algorithm is developed to construct a discrete scale space feature sketch from a set of multi-scale features.
- An algorithm for performing characteristic scale selection using the scale space feature sketch.
- An evaluation is performed to compare the new scale selection technique to the existing state of the art method, when used as part of various types of feature extractors. The evaluation shows that the new technique is superior to the existing technique in several ways.

Affine Adaptation

A novel affine adaptation method is presented that employs the Hessian matrix to estimate the shape of local features. To this end, the following contributions are made:

- Theoretical derivations show that the Hessian matrix can be used to measure the shape of image intensity distributions.
- The Hessian matrix-based iterative affine adaptation algorithm.
- An evaluation comparing the Hessian matrix to the second moment matrix (the existing state of the art method for scale selection). The evaluation shows that the Hessian matrix-based method achieves superior computational efficiency and equivalent feature quality when combined with a blob-like feature extractor. When combined with a corner extractor, the Hessian matrix-based method is more efficient, but produces features of inferior quality, compared to the second moment matrix.

Second Tier Correspondence Extraction Methods

Several novel methods are presented that refine the quality and accuracy of correspondences, and extract additional correspondences after the traditional wide baseline matching has been performed. Contributions include:

- An alignment method for accurately aligning matched affine features and rejecting mismatches.
- A method for producing additional matches by exploring the local vicinity of pairs of aligned features. The method makes use of the information

contained in the aligned pair of features to align neighbouring regions in the images.

- A method for extracting highly accurate, small scale point correspondences from a pair of aligned features.

- The above methods are combined to form a system, referred to as WiDense, that extracts large numbers of highly accurate correspondences. This system does not require knowledge of the camera or scene geometry, but only requires a set of putative affine correspondences. Even when the input correspondences are not sufficient to compute the camera geometry, the WiDense system can produce many additional correspondences.

- An evaluation of the WiDense system. The evaluation shows that in scenarios where existing wide baseline matching methods fail to produce adequate correspondences to compute the camera geometry, the additional correspondences generated by by the WiDense system can assist in computing the geometry. The addition of WiDense generated correspondences doubled the success rate of computing the epipolar geometry in a range of very challenging scenarios.

Evaluation Method

A new evaluation method is contributed, which measures how useful a correspondence extraction system is in a practical epipolar geometry computation task. The evaluation operates on high resolution image sets taken from a pair of cameras. Ground truth data and test images are generated automatically from the high resolution image sets. Results are presented in terms the average proportion of trials where the epipolar geometry was computed with sufficiently low error. A dataset containing six sets of images was compiled. The sets were captured with the cameras in a different configuration such that the difficulty of computing the geometry ranged from hard to nearly impossible (given existing methods).

1.5 Publications Resulting from Research

1.5.1 Submitted to International Journals

R. Lakemond, C. Fookes, and S. Sridharan, "WiDense – a method for extracting dense optical flow across uncalibrated wide baseline views," submitted to *International Journal of Computer Vision* in December 2009.

R. Lakemond, C. Fookes, and S. Sridharan, "Affine adaptation using the hessian matrix," submitted by invitation to *EUROASIP Journal on Image and Video Processing Special Issue on "Advanced Video-Based Surveillance"* in January 2010.

1.5.2 International Conferences

R. Lakemond, D. McKinnon, C. Fookes, and S. Sridharan, "A feature clustering algorithm for scale-space analysis of image structures," in *Proc. Signal Processing and Communication Systems, International Conference on*, pp. 186–192, 2007.

D. McKinnon, **R. Lakemond**, C. Fookes, and S. Sridharan, "Ground-plane based projective reconstruction for surveillance camera networks," in *Proc. Signal Processing and Communication Systems, International Conference on*, pp. 423–428, 2007.

R. Lakemond, C. Fookes, and S. Sridharan, "Affine adaptation of local image features using the hessian matrix," in *Proc. Advanced Video and Signal Based Surveillance, IEEE conference on*, pp. 496–501, 2009.

R. Lakemond, C. Fookes, and S. Sridharan, "Dense correspondence extraction in difficult uncalibrated scenarios," in *Proc. Digital Image Computing: Techniques and Applications*, pp. 53–60, 2009.

R. Lakemond, C. Fookes, and S. Sridharan, "Phd forum: multiple camera management using wide base-line matching," in *Proc. Distributed Smart Cameras, International Conference on*, 2009.

J. Liu, D. ORourke, T. Wark, **R. Lakemond**, and S. Sridharan, "Camera calibration in wireless multimedia sensor networks," in *Proc. Intelligent Sensors, Sensor Networks and Information Processing, International Conference on*, pp. 301–306, 2009, (best student paper award).

Chapter 2

Projective Geometry and Camera Calibration

Projective geometry is the study of geometric properties which are invariant under projective transformations. It encompasses camera geometry and the formation of images through 3D to 2D projection. The algebra of projective geometry serves to formally express the concepts of projective geometry, and is used in the design of image processing algorithms.

This chapter introduces basic concepts in projective geometry that are used throughout this thesis. The subject will be treated in a manner that provides the understanding and algebraic tools required to formulate and solve the problem of extracting correspondences between different views. Geometric invariance properties will not be discussed in great detail. The focus will instead be on the practical implications for image processing. For a more detailed treatment of projective geometry, the interested reader is referred to [22].

The following section introduces basic concepts and notation. Section 2.2 defines various coordinate transformations and their application to image processing. The pinhole camera model is introduced in Section 2.3, along with a brief discussion on lens distortion models. In Section 2.4 the concept of epipolar geometry is explained and in Section 2.5 some approaches to camera calibration are explored. The chapter is concluded in Section 2.6.

2.1 Homogeneous Representation of Points and Lines

A line in a 2D plane, $\mathbb{R}^2$, may be represented by an equation, $ax+by+c=0$, or by a vector of its parameters, $\mathbf{l} = [a\ b\ c]^\top$. Multiplying this vector by any non-zero constant generates an equivalent line vector, since the line equation represents the same line. The class of vectors subject to this equivalence relationship is known as homogeneous vectors.

The line equation can be expressed as the inner product of a vector containing the point coordinates and the line vector, $[x\ y\ 1]\,[a\ b\ c]^\top = 0$. Multiplying by a non-zero constant gives an equivalent expression, $[kx\ ky\ k]\,[a\ b\ c]^\top = 0$. The vector $[x\ y\ 1]^\top$ is therefore a homogeneous vector representation of the coordinates $[x\ y]^\top$ in $\mathbb{R}^2$. The representation, $\mathbf{p} = [x\ y\ w]^\top$, is referred to as the homogeneous coordinate representation of the inhomogeneous point $\mathbf{p} = \left[\frac{x}{w}\ \frac{y}{w}\right]^\top$.

The equivalence of $[x\ y\ 1]^\top$ and $[kx\ ky\ k]^\top$ in projective geometry may be modelled in terms of the projection of a point in $\mathbb{R}^3$ through the coordinate origin to a plane parallel to the $x-y$ plane and intersecting the z axis at $z=1$. The ray passing through the origin and the point $k\mathbf{p} = [kx\ ky\ k]^\top$ also passes through the point $\mathbf{p} = [x\ y\ 1]^\top$ on the projection plane. These two points are therefore equivalent since they correspond to the same projection ray and are indistinguishable in terms of the projection plane.

The coordinate space in which the homogeneous point, $\mathbf{p} = [x\ y\ w]^\top$, and line $\mathbf{l} = [a\ b\ c]^\top$ are defined is referred to as the projective two space, $\mathbb{P}^2$. Similarly, projective three space, $\mathbb{P}^3$, may be defined with homogeneous points, $\mathbf{p} = [x\ y\ z\ w]^\top$, lines, $\mathbf{l} = [a\ b\ c\ d]^\top$ and planes, $\boldsymbol{\pi} = [a\ b\ c\ d]^\top$ (represented by their normal vectors).

The homogeneous vector representation has several convenient characteristics. Firstly, points at infinity are simply represented as $\mathbf{p} = [x\ y\ 0]^\top$. The equivalent inhomogeneous point, $\left[\frac{x}{0}\ \frac{y}{0}\right]^\top$, has no real meaning, but the homogeneous representation may be used meaningfully in algebra and is critical to camera calibration. Secondly, linear coordinate transformations may be represented as a simple matrix multiplication. For example, in inhomogeneous coordinates a point $\mathbf{p} = [x\ y]^\top$ may be scaled by factor a and translated by $\mathbf{t} = [t_x\ t_y]^\top$ as,

$$\mathbf{p}^{'} = a\mathbf{p} + \mathbf{t}.$$

In homogeneous coordinates, this may be written as,

$$\mathbf{p}^{'} = \begin{bmatrix} a & 0 & t_x \\ 0 & a & t_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}.$$

Linear transformations are the subject of the following section.

2.2 Projective Transformations

This section defines various transformations of $\mathbb{P}^2$ and discusses their use and implementation for image processing. The class of transformations that will be discussed is referred to as a projective transformation, a projectivity, or a homography. A projective transformation, $H(\mathbf{x})$ of homogeneous coordinates $\mathbf{x}$ has the following properties:

1. H is a mapping from $\mathbb{P}^n$ to itself, $H : \mathbb{P}^n \rightarrow \mathbb{P}^n$.
2. H is a linear mapping, such that it may be expressed as multiplication with a non-singular matrix, $H(\mathbf{x}) = \mathbf{H}_{n\times n}\mathbf{x}$, where $\mathbf{H}_{n\times n}$ is a homogeneous matrix (defined up to an arbitrary non-zero scale factor). The mapping is therefore also invertible.
3. If any three points, $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are collinear, then the transformed points, $\mathbf{H}\mathbf{x}_1$, $\mathbf{H}\mathbf{x}_2$ and $\mathbf{H}\mathbf{x}_3$, are also collinear.
4. The composition of two projective transformations produces another projective transformation, $H_1(H_2(\mathbf{x})) = \mathbf{H}_1\mathbf{H}_2\mathbf{x} = \mathbf{H}_{12}\mathbf{x}$.
5. Because the coordinate system is homogeneous, transformation matrices are also homogeneous – $k\mathbf{H}$ is equivalent to $\mathbf{H}$ for any constant, non-zero k.

These transformations can be used to describe the relationship between different projected images in several ways. If two images are produced by projecting through the same point onto two different planes, then the images are related by a projective transformation. Similarly, a planar surface projected through different viewpoints is related to each image by a projective transformation, and hence the images are also related to each other by a projective transformation. Two projections, through different viewpoints, of a surface that is not planar are not related by a projective transformation.

2.2.1 2D Projective Transformations

In $\mathbb{P}^2$, a general projective transformation $\mathbf{H}$ has eight degrees of freedom. It is possible to parameterise a projective transformation such that each parameter represents a degree of freedom and a distinct geometric effect. A transformation may also be decomposed into a series of transformations so that each one addresses a particular geometric effect. This section presents each of these geometric effects as a separate transformation before combining them into a general projective transformation. Transformation matrices may be expressed in terms of their parameters, for example $\mathbf{M}(m_1, m_2, \ldots, m_n)$, or simply by their symbol, $\mathbf{M}$, if the parameters have previously been defined.

Translation

A translation transformation may be defined as,

$$\mathbf{T}(t_x, t_y) = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}.$$

A translation simply shifts all points in the same direction, $\mathbf{Tx} = \mathbf{x} + \mathbf{t}$. The inverse translation is the translation in the opposite direction, $\mathbf{T}^{-1}(t_x, t_y) = \mathbf{T}(-t_x, -t_y)$.

Rotation

A rotation around the coordinate origin may be defined as,

$$\mathbf{R}(\theta) = \begin{bmatrix} \mathbf{R}' & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The inverse rotation is the rotation in the opposite direction, $\mathbf{R}^{-1}(\theta) = \mathbf{R}(-\theta)$.

In order to rotate the coordinates around a point other than the origin, the point must first be translated to the origin, then the rotation must be applied and lastly the point must be translated back from the origin to its original location, $\mathbf{R}(\theta, c_x, c_y) = \mathbf{T}(c_x, c_y)\,\mathbf{R}(\theta)\,\mathbf{T}(-c_x, -c_y)$.

Scaling

A scaling transformation is defined as,

$$\mathbf{K}(k) = \begin{bmatrix} \mathbf{K}' & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Note that scaling does not consist of simply multiplying a point vector by a constant. Because homogeneous vectors are used, such an operation generates an equivalent point vector. On the other hand, $\mathbf{K}(k)\,\mathbf{x} = (kx, ky, w)^\top$ results in scaled coordinates. The inverse scaling is, $\mathbf{K}^{-1}(k) = \mathbf{K}(k^{-1})$.

Anisotropic Scaling

An anisotropic scaling transformation scales coordinates by different amounts in two orthogonal directions. The scaling introduced in the previous section, in contrast, is referred to as isotropic scaling, since it affects coordinates to the same degree in all directions. An anisotropic scaling transformation may be defined as,

$$\begin{aligned}\mathbf{A}\left(q,\phi\right) &= \begin{bmatrix}\mathbf{A}' & \mathbf{0}\\ \mathbf{0}^\top & 1\end{bmatrix} = \mathbf{R}\left(-\phi\right)\mathbf{D}\left(q\right)\mathbf{R}\left(\phi\right),\\ \mathbf{D}\left(q\right) &= \begin{bmatrix} q & 0 & 0\\ 0 & q^{-1} & 0\\ 0 & 0 & 1\end{bmatrix}.\end{aligned}$$

This transformation may be interpreted as a rotation, followed by an anisotropic scaling along the coordinate axes, and finally a counter rotation. The axis of scaling is specified by parameter ϕ. The ratio of the amount of scaling along each orthogonal direction is q^2. Note that in the above definition, the determinant of the matrix $\mathbf{D}$ is one. This ensures that, when a transformation contains both isotropic and anisotropic scaling, a distinction can be made between the two components and so that their parameters each represent a unique degree of freedom. The inverse is simply, $\mathbf{A}^{-1}\left(q,\phi\right) = \mathbf{A}\left(q^{-1},\phi\right)$.

Projective Scaling

Projective scaling models the perspective effect and is defined as follows,

$$\mathbf{P}\left(v_1,v_2\right) = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{v}^\top & 1\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0\\ 0 & 1 & 0\\ v_1 & v_2 & 1\end{bmatrix}.$$

Transforming a point using $\mathbf{P}$ gives $\mathbf{Px} = (x, y, v_1 x + v_1 y + 1)$, or equivalently,

$$\mathbf{Px} = \left(\frac{x}{v_1 x + v_1 y + 1}, \frac{y}{v_1 x + v_1 y + 1}, 1\right).$$

This transformation introduces scaling that varies with a change in coordinates.

Reflection

A reflection negates one of the coordinates. The two reflections are defined as,

$$\mathbf{N}_x = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{N}_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Reflections will not be used in this thesis.

Transformation Classes

Transformations are classified according to invariance properties – the geometric properties that are preserved after the transformation has been applied. These classifications are used to distinguish between different types of transformations.

A Euclidean transformation, or isometry, consists of rotation and translation (three parameters),

$$\mathbf{M} = \begin{bmatrix} \mathbf{R}' & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}.$$

A similarity transformation consists of rotation, translation and isotropic scaling (four parameters),

$$\mathbf{S} = \begin{bmatrix} \mathbf{R}'\mathbf{K}' & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}.$$

An affine transformation includes anisotropic scaling (six parameters),

$$\mathbf{H} = \begin{bmatrix} \mathbf{A}'\mathbf{R}'\mathbf{K}' & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}.$$

A projective transformation includes projective scaling (eight parameters),

$$\mathbf{P} = \begin{bmatrix} \mathbf{A}'\mathbf{R}'\mathbf{K}' & \mathbf{t} \\ \mathbf{v}^\top & 1 \end{bmatrix}. \tag{2.1}$$

A projective transformation may also be defined as,

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}^{'} & \mathbf{t} \\ \mathbf{v}^{\top} & 0 \end{bmatrix}.$$

In this case, decomposition into separate components is not possible as it is in equation 2.1.

2.2.2 Image Transformation

One may state that point $\mathbf{x}$ is mapped to point $\mathbf{x}^{'}$ by transformation $\mathbf{H}$, according to $\mathbf{x}^{'} = \mathbf{H}\mathbf{x}$. Digital images consist of intensity values measured at a set of discrete sampling points. When transforming images, it seems natural to express the transformation as the transfer of the intensity value at point $\mathbf{x}_i$ to the point $\mathbf{x}_i^{'} = \mathbf{H}\mathbf{x}_i$ for all i. This usually results in the intensity samples being transferred to coordinates that are not on a convenient sampling grid. A more practical approach is to map the sampling points of the transformed image to the source image using the inverse transformation, and using bilinear interpolation to compute the new samples. The transformation of image $I(\mathbf{x})$ to image $I^{'}(\mathbf{x})$ using transformation matrix $\mathbf{H}$ is expressed as follows,

$$I^{'}(\mathbf{x}) = I(\mathbf{H}\mathbf{x}). \tag{2.2}$$

Here $\mathbf{x}$ can be any point (or all points) within the domain of image $I^{'}(\mathbf{x})$. It is possible that $\mathbf{H}\mathbf{x}$ falls outside the domain of image $I(\mathbf{x})$. A common approach to dealing with this situation is to assign a background intensity value (usually zero) to $I^{'}(\mathbf{x})$ for all values of $\mathbf{x}$ where $\mathbf{H}\mathbf{x}$ falls outside image $I(\mathbf{x})$.

2.3 The Pinhole Camera Model

Cameras project rays of light from the 3D world onto a 2D image plane. The action of a camera is commonly modelled using the pinhole camera model, or the central projection camera model. In this model, all projection rays pass through a single point, the camera centre. In reality, cameras have a significantly large aperture and a lens that deflects light. A high quality lens may produce an image that is very similar to an equivalent pinhole camera. The distortions introduced by lenses may be accounted for by a non-linear mapping between distorted lens coordinates and undistorted, or ideal projection coordinates. This thesis will make use of the pinhole camera model only and assume that lens distortion is negligible.

The pinhole camera model consists of a 3×4 matrix that projects 3D homogeneous points to 2D homogeneous points. The model is defined as,

$$\begin{aligned} \mathbf{P} &= \mathbf{KR}\left[\,\mathbf{I}\,\middle|\,-\mathbf{c}\,\right] = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}, \\ \mathbf{K} &= \begin{bmatrix} fm_x & s & x_c \\ 0 & fm_y & y_c \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

Here $\mathbf{c}$ is the position of the camera centre in the world coordinate frame ($\mathbb{P}^3$), $\mathbf{R}$ is a 3×3 rotation matrix representing the orientation of the camera coordinate frame, and $\mathbf{K}$ is referred to as the camera calibration matrix. The elements in $\mathbf{K}$ are referred to as the camera intrinsic parameters. f is the focal length – the distance of the image plane to the camera centre. m_x and m_y are the number of pixels per unit distance in the image. In most cases the camera pixels are square, so that $m_x = m_y$, and they are often chosen to equal 1, so that f is expressed in terms of pixels. s is the skew parameter and is zero for most cameras (with orthogonal pixel arrays). $(x_c, y_c)^\top$ is the principal point – the point, in pixel coordinates, where the line joining the camera centre and the image plane (known as the principal axis) is perpendicular to the image plane.

The action of the camera on world point $\mathbf{x}_3 \in \mathbb{P}^3$ to produce image point $\mathbf{x}_2 \in \mathbb{P}^2$ is expressed simply as,

$$\mathbf{x}_2 = \mathbf{P}\mathbf{x}_3.$$

2.4 Epipolar Geometry

Epipolar geometry describes the geometry of two cameras or views. It depends on the intrinsic parameters and relative orientation of the camera, but is independent of scene geometry. Epipolar geometry is conveniently represented by the fundamental matrix, $\mathbf{F}$, a 3×3 matrix with rank two. The properties of epipolar geometry will be explored in terms of the fundamental matrix and two cameras, $\mathbf{P}$ and $\mathbf{P}^{'}$ with camera centres $\mathbf{c}_3$ and $\mathbf{c}_3^{'}$.

The baseline of two cameras is the line joining the camera centres. The epipoles, $\mathbf{e}$ and $\mathbf{e}^{'}$ are the intersections of the baseline with the image planes. Alternatively, epipole $\mathbf{e}$ is the image of camera centre $\mathbf{c}_3^{'}$ as viewed from $\mathbf{P}$, $\mathbf{e} = \mathbf{P}\mathbf{c}_3^{'}$. Similarly, $\mathbf{e}^{'} = \mathbf{P}^{'}\mathbf{c}_3$.

Any plane that contains the baseline is known as an epipolar plane. The intersection of an epipolar plane with an image plane is an epipolar line. An epipolar plane may be defined by a point in space, $\mathbf{x}_3$ and the two camera centres. This plane produces epipolar lines $\mathbf{l}$ and $\mathbf{l}^{'}$ in each image. The points $\mathbf{x}_2$ and $\mathbf{x}_2^{'}$ that are the images of $\mathbf{x}_3$ lie on these epipolar lines, so that $\mathbf{x}_2^\top \mathbf{l} = \mathbf{x}_2^{'\top} \mathbf{l}^{'}$.

The fundamental matrix provides a mapping from the image of a point in one

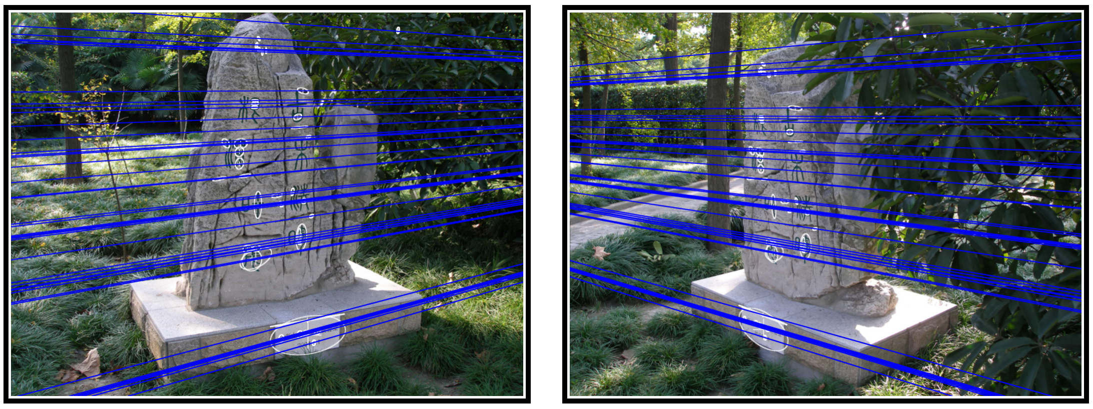


Figure 2.1: A pair of views of the same scene with epipolar lines of selected points overlayed in blue.

view to the corresponding epipolar line in the other view, $\mathbf{l} = \mathbf{F}^\top \mathbf{x}_2^{'}$ and $\mathbf{l}^{'} = \mathbf{F}\mathbf{x}_2$. Figure 2.1 illustrates this property – for each point marked with a white cross and ellipse, a blue epipolar line is overlayed on the other image, where it intersects the corresponding point. From this property, it is found that $\mathbf{x}_2^{'\top}\mathbf{F}\mathbf{x}_2 = 0$. This property is particularly useful when searching for correspondences, since it allows the search space to be restricted to the vicinity of the epipolar line.

All epipolar lines intersect the epipole, $\mathbf{e}^{'\top}\mathbf{l}^{'} = 0$, therefore $\mathbf{e}^{'\top}\mathbf{F}\mathbf{x}_2 = 0$ for all $\mathbf{x}_2$. Therefore $\mathbf{e}^{'\top}\mathbf{F} = 0$ and similarly $\mathbf{e}^\top\mathbf{F}^\top = \mathbf{F}\mathbf{e} = 0$. $\mathbf{e}^{'}$ is the left null-vector of $\mathbf{F}$ and $\mathbf{e}$ is its the right null-vector.

One of the most important properties of the fundamental matrix is that it can be used to define a pair of camera matrices as,

$$\mathbf{P} = \left[\ \mathbf{I} \ \middle|\ \mathbf{0}\ \right], \quad \mathbf{P}^{'} = \left[\ [\mathbf{e}^{'}]_\times \mathbf{F} \ \middle|\ \mathbf{e}^{'}\ \right]. \tag{2.3}$$

The above cameras are related to the real cameras by a projective transformation. When reconstructing cameras and scenes from correspondences between two cameras in general position, it is only possible to produce a reconstruction with projective ambiguity – the reconstruction differs from the real system by a projective transformation which cannot be resolved without more information about the scene. The reconstruction in equation 2.3 is therefore a most useful result. The fundamental matrix is often used in the above way as the first step in reconstructing cameras and scenes.

2.5 Camera Calibration

The camera matrix for any camera can be computed from a set of corresponding 3D and image points or lines by means of the direct linear transformation algorithm and appropriate data normalisation. The most difficult part of this process is acquiring the correspondences. Measurements must be taken in the 3D world and corresponding points or lines marked in the image. This is a largely manual process. A common method for acquiring accurate world coordinate measurements is to construct a calibration object with many regularly spaced points. A manual or image processing method is then used to locate the corresponding points in images. These methods all require a significant amount of labour on the part of an operator or technician for each camera.

It is possible to extract camera models purely from correspondences between multiple camera views. Unfortunately, without real world measurements, the absolute position, orientation and scale of a set of cameras can not be recovered. Only the relative camera positions, orientations and calibration matrices can be found. In the case where only two cameras are available, the reconstruction will contain projective ambiguity – that is, the reconstruction will be related to the real cameras by an unknown projective transformation. A reconstruction up to a similarity or projective transformation is, in many cases, sufficient in order for multiple camera computer vision techniques to effectively transfer information between views.

It is therefore possible to compute a sufficient camera calibration model using only correspondences between views. The calibration procedure may by automated by automating the process of extracting correspondences between views. Automatic correspondence extraction is a difficult problem if the cameras are placed in a sparse arrangement, as they often are in surveillance scenarios. This is the motivation for the research presented in this thesis – better methods for extracting correspondences in difficult scenarios are sought in order to address the requirements for automatic calibration.

2.6 Chapter Summary

This chapter presents the basic tools of projective geometry that will be used throughout this thesis to express and solve problems. The concepts of homogeneous representation of coordinates, lines, planes and transformations are introduced. Various types of linear projective transformations are described. These will be used extensively throughout this thesis to represent features and local image relations.

The central projection camera model, epipolar geometry and camera calibration are briefly introduced. These concepts form the basis of understanding camera operation and relationships between cameras and provide the motivation for developing automatic correspondence extraction techniques.

Chapter 3

The Wide Baseline Matching Problem

Image alignment or image registration is one of the fundamental problems in image processing and computer vision. It is an essential step in any system that attempts to make some comparison between sets of visual information, extract meaningful measurements from images, or combine visual information from multiple sources. Automatic calibration aims to align multiple views of a scene by means of a model of the cameras and scene. When the baseline (the line joining camera centres) is relatively large, then finding the alignment can be very challenging. This chapter examines the problem of aligning wide baseline image pairs and broadly examines wide baseline matching as a method for addressing this problem.

Section 3.1 defines the wide baseline image alignment problem. Feature-based alignment is presented in broad terms in Section 3.2. The discussion focuses on the wide baseline scenario, though feature-based methods are suitable for many applications. Methods for evaluating the performance of local feature extractors are reviewed in Section 3.3. Section 3.4 reviews the published literature pertaining to the feature extraction, description and matching techniques relevant to wide baseline matching. The chapter is concluded in a summary in Section 3.5.

3.1 The Wide Baseline Matching Problem

A picture may be worth a thousand words, but it captures only a fraction of the information in a scene. Images captured from different view points capture different information from the scene. By combining the information from multiple 2D views it is possible to recover some of the 3D world and camera configuration information. It is only possible to combine information from different views if the relationship between the views and the relationship of related image segments can be discovered. These are the fundamental problems in computer vision – alignment and segmentation.

The images of two cameras viewing the same scene appear different due to the different relative arrangement of objects in the scene. The difference in appearance is limited if the two cameras are placed close together. Increasing the distance between cameras, or the baseline, results in complicated changes in appearance that are unpredictable without extensive knowledge of the scene and cameras. The two images cannot be aligned by a linear transformation of image coordinates, unless the scene consists of only one planar surface.

Apart from the effects of projection, the variation in appearance between widely separate views may be influenced by the following factors:

- Camera lens distortion (non-linear).
- Different photometric responses of the cameras (as a result of vignetting or

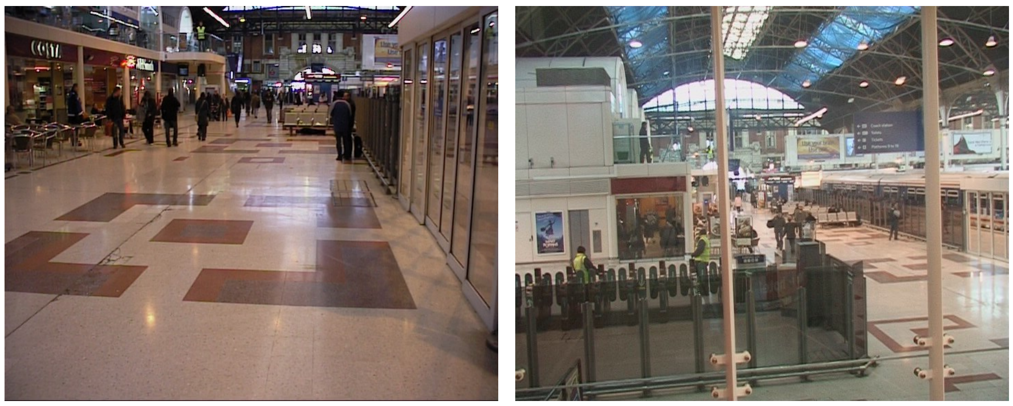


Figure 3.1: Images from two of the surveillance cameras in the PETS2006 database. The two viewpoints are widely separated, but share some overlap.

automatic gain control, for example).

- Specular reflections (reflection of bright light sources off some surfaces).

There is often no prior knowledge of how any of the above effects are presented in a particular case. Figure 3.1 shows an example of two cameras in a typical surveillance network with some overlapping view (sourced from `http://www.cvg.rdg.ac.uk/PETS2006/data.html`). This example exhibits many of the characteristics listed above. It is difficult, even for a human, to find the corresponding regions between these two images. The task of aligning such views automatically poses a great challenge.

The relationship between widely separated views can only be fully expressed in terms of a geometric model that captures the 3D to 2D projection, such as epipolar geometry or calibrated camera models and 3D scene models (Chapter 2). The complex relationship between views makes it impossible to align the images by comparing them directly. However, it is feasible to align local subregions of the images that correspond to small, continuous surface sections in the 3D scene. The alignment problem is considerably simpler in a local sense than the global alignment problem. The following approximations may be used to simplify the alignment of a small local area:

- Lens distortion is locally negligible (except for very wide angle lenses, not treated in this thesis).
- The photometric relationship is approximately linear over small corresponding regions.
- Surfaces are locally planar. Different projections of a small local surface are therefore related by a linear transformation and may even be approximated by an affine transformation.
- The relative order of elements in a local area is preserved across views.
- Occlusions and regions affected by specular reflections may be treated as binary – a particular region is either occluded (and not observable) or not occluded.

It is possible to align small local regions individually across views, and to thereby collect a set of corresponding regions. An appropriate geometric model relating the two views may be computed using the set of correspondences. The wide baseline image alignment problem is thereby reduced to a problem of finding corresponding local regions – the wide baseline matching problem.

3.2 Matching using Local Features

Despite the assumptions and approximations that may be applied when matching small local regions, wide baseline matching is still a difficult task. There are several issues that need to be taken into consideration in addressing the problem.

Firstly, it is not possible to align any and all regions in an image. A region must contain sufficient information to identify it uniquely across multiple views. It must also be well localised spatially and its scale and shape must be well defined. Regions that occupy only a few pixels in the image, for example, do not contain sufficient information to distinguish them from all other regions. Regions of homogeneous intensity do not provide a means for accurate localisation and cannot be accurately aligned, even if they can be matched. It is therefore necessary to carefully select the regions to align.

Local image feature extraction provides a method to deal with the problem of selecting regions for matching. Local image features are patterns in an image that are defined in limited image areas and are distinguishable from the surrounding image in some way. Various types of local features may be found in images, including line segments, contours, corners and blobs (examples include [17, 32, 49]). Local features are spatially well defined – they have a location and spatial extent. Global image features, in contrast, are a representation of information in the entire image.

The next problem is to establish correspondences between selected features. With proper normalisation it becomes possible to compare feature regions directly through pixel differences. Directly comparing normalised regions not only requires a large amount of memory and computation time, but it does not yield good results. Feature descriptors provide a method for representing features in a more compact manner and at the same time make the matching process more robust. A descriptor is a vector representation computed from the feature image region.

Matching features using descriptors usually results in a set of putative correspondences that is contaminated by incorrect matches. In difficult wide baseline scenarios there may be more incorrect matches than correct matches. The geometric model parameters may be computed from the set of putative correspondences by means of a robust parameter estimation method such as Random Sample Consensus (RANSAC) [19] or Least Median of Squares (LMS) [50].

In summary, wide baseline matching, or feature-based alignment in general, follows this procedure: First regions that are good candidates for alignment are selected – the local features. Each local feature is assigned a descriptor based on the feature appearance. Sets of features are then compared across views by means of their descriptors, in order to select matching features. The scene geometry, in terms of some parametric model, is recovered from the set of putative correspondences using a robust parameter estimation method.

The rest of this section introduces basic and general concepts relevant to local feature-based alignment. The remainder of the chapter reviews the published literature on the topic.

3.2.1 Terminology

The computer vision literature makes use of a large number of terms to describe various concepts related to local features. Ambiguities and inconsistencies in terminology are common. This section presents the terminological conventions used throughout this thesis.

Feature. Used to indicate *local image feature.* This thesis is concerned with local image features only. Referring to these as simply features is therefore unambiguous. A number of alternative terms appear in the literature, including *interest point* or *point of interest* and *region of interest* (ROI). In practice, a feature consists of a support region that defines the feature (see below). A feature is therefore interpreted as the information required to isolate the feature support region.

Feature support region. Features generally have a local image region associated with them, either because this region defines the structure of the feature or because this region influenced the feature extractor to select the feature. This is referred to as the support region in this thesis. Some types of features, such as corners, may ideally be represented by a point – a geometric entity with a defined location, but no spatial extent. In practice, images are discrete and the majority of the algorithms used to extract features must process a significant region of the image in order to extract a feature. Most practical features therefore at least have a finite spatial extent, defining its support region. The centre point of this region can usually be employed by geometric algorithms as if the feature were a point structure.

Feature extractor. An algorithm or its implementation that processes an image to detect the presence of local image features and extract these features. The term *feature detector* is also frequently used with the same meaning. Feature extractor is used in this thesis since most of the relevant algorithms extract a significant amount of information in conjunction with each feature, rather than merely detecting the presence of features.

Descriptor. A vector representation used to characterise local features so that they may be matched to corresponding features and distinguished from dissimilar features. A descriptor is computed from the feature support region.

Covariance. If a feature extractor is covariant under a specific type of transformation, then if the feature extractor is applied to two images related by such a transformation, the corresponding features it produces are related according to the same transformation. In other words, if two images are related according to $I_2(\mathbf{x}) = I_1(\mathbf{Hx})$, and the extractor is covariant to the class of transformation that $\mathbf{H}$ belongs to, then corresponding features extracted from these two images will be related according to $\mathbf{F}_2 = \mathbf{HF}_1$ (see Section 3.2.2 below for a discussion on representing features by normalisation transformations). This information may be used to extract invariant descriptors (see below).

Invariance. If a feature extractor is invariant under a specific type of transformation, then if the feature extractor is applied to two images related by such a transformation, it will produce the same features from both images. In other words, if two images are related according to $I_2(\mathbf{x}) = I_1(\mathbf{Hx})$, and the extractor is covariant to the class of transformation that $\mathbf{H}$ belongs to, then corresponding features extracted from these two images will be related according to $\mathbf{F}_2 = \mathbf{F}_1$.

A covariant pair of corresponding features provide a means for normalising their support regions so that the normalised regions are invariant. If $I_2(\mathbf{x}) = I_1(\mathbf{Hx})$ and $\mathbf{F}_2 = \mathbf{H}^{-1}\mathbf{F}_1$, then $I_2(\mathbf{F}_2\mathbf{x}) = I_1(\mathbf{F}_1\mathbf{x})$ (at least to some local extent). This is explained in more detail in the next section.

3.2.2 Feature Representation and Description

As defined in the previous section, a feature is defined as the information required to isolate the feature support region. In this thesis, features are represented by an affine transformation, $\mathbf{F}$, that represents a mapping from a unit circle centred at the origin to an ellipse circumscribing the feature support region. The feature transformation is composed as,

$$\mathbf{F}(t_x, t_y, q, \phi, \theta, k) = \mathbf{T}(t_x, t_y)\,\mathbf{A}(q, \phi)\,\mathbf{R}(\theta)\,\mathbf{K}(k)\,. \tag{3.1}$$

The component transformations that make up $\mathbf{F}$ are defined in Chapter 2, Section 2.2.1. The parameters of each feature's transformation are determined by the feature extractor. Not all feature extractors produce features with full affine transformations, but all extractors at least provide the translation component (which gives the feature location).

The feature transformation can be used to sample a normalised image for the purpose of computing the descriptor. The normalised image is computed as,

$$I_n(\mathbf{x}) = I\left(\mathbf{F}\mathbf{K}\left(k_d^{-1}\right)\mathbf{x}\right), \quad x, y \in [-k_d, k_d]\,, \tag{3.2}$$

where k_d is the desired resolution of the normalised image. Feature descriptors are computed from the normalised images of features because the normalisation accounts for some of the local variations in image appearance.

If an affine covariant feature extractor is used to extract two corresponding features, $\mathbf{F}_1$ and $\mathbf{F}_2$, from a pair of images, $I_1(\mathbf{x})$ and $I_2(\mathbf{x})$, related by an affine transformation $I_1(\mathbf{x}) = I_2(\mathbf{H}_A\mathbf{x})$, then by definition the features are related by the same transformation that relates the images, $\mathbf{F}_1 = \mathbf{H}_A^{-1}\mathbf{F}_2$. A consequence of this fact is that the normalised images computed from each feature are the same and hence the descriptors computed from these normalised images are the same. A covariant feature therefore allows the computation of an invariant descriptor. In practice the features will not be perfectly covariant, since they are computed independently from different images with noise. This results in some variation in the normalised images and descriptors.

3.3 Evaluation Methods

Two evaluations of affine covariant feature extractors have been published, each with its own evaluation method. These two methods are described in this section. The relevant results of the two evaluations are discussed as part of the review of feature extractors in Section 3.4.1.

3.3.1 Feature Repeatability Test

The affine feature extractor evaluation method presented in [45] computes four metrics – repeatability, correspondence count, matching score and the number of correct matches. An image dataset is provided in which the images are related by planar projective transformations. The test metrics are computed using the knowledge of the relationship between images.

The repeatability metric is the affine extension to the point correspondence repeatability introduced in [52]. An overlap error is defined to measure the amount of error in relative position and shape of corresponding features detected in different views. Overlap is defined as,

$$\epsilon_o = 1 - \frac{E\left(\mu_a\right) \cap E\left(\mathbf{H}^\top \mu_b \mathbf{H}\right)}{E\left(\mu_a\right) \cup E\left(\mathbf{H}^\top \mu_b \mathbf{H}\right)}.$$

Here $E\left(\mu\right)$ defines a region bound by the elliptic curve $\mathbf{x}^\top \mu \mathbf{x} = 1$. The ellipse is derived from the affine feature shape and is normalised to have an average radius of 30 pixels. The normalisation is necessary because the overlap error of features could otherwise be reduced by simply enlarging the features. $\mathbf{H}$ is the homography mapping image b to image a, so that $\mu_1 = \mathbf{H}^\top \mu_2 \mathbf{H}$, if μ_1 and μ_2 are exactly corresponding elliptic curves in images a and b respectively. $E\left(\mu_a\right) \cap E\left(\mathbf{H}^\top \mu_b \mathbf{H}\right)$ is the union of the regions and $E\left(\mu_a\right) \cup E\left(\mathbf{H}^\top \mu_b \mathbf{H}\right)$ is the intersection. The intersection of the regions is computed numerically. Features with overlap error below 40% are considered repeated and are counted as correspondences. The repeatability measure is defined as the number of correspondences divided by the minimum number of features in the common parts of the two images.

The matching score and number of correct matches give an indication of how well the features can be matched using a given descriptor. Features are matched using their descriptors and one-to-one nearest neighbour matching. Matches are deemed correct if the overlap error of a given match is below 40%. The matching score is the number of correct matches divided by the minimum number of features in the common part of the two images.

The dataset consists of eight sets of images, accompanied by a set of transformations relating the test images to the reference image in each set. Each set of images exhibits a variation of one property. There are two image sets that contain a change in viewpoint, two that contain a change in scale, two that contain varying blur (camera focus), one that contains varying JPEG compression and one with varying camera exposure time. The test data, ground truth homographies and test procedure scripts are readily available from `http://www.robots.ox.ac.uk/~vgg/data/data-aff.html`. The main shortcoming of this evaluation method is the small number of test images in the dataset.

This evaluation method is used to evaluate some of the new feature extraction methods presented in this thesis. An efficiency metric was added to measure the computational efficiency with which a feature extractor produces correct matches. The efficiency is expressed in terms of the rate, r, at which matches are produced,

$$r = \frac{c_i}{t_i + t_0},$$

where c_i is the number of correct matches between reference image 0 and test image i, and t_i and t_0 are the times taken to extract features from image i and 0, respectively.

3.3.2 3D Object-based Evaluation

In [47], an evaluation is presented that uses a database of 100 objects, photographed on a turntable at 5° rotation intervals, in three different lighting conditions, using two cameras. The evaluation operates by extracting features from all of the images in the database, as well as a set of unrelated images (to generate background interference features). Sample features are then matched to the entire database using nearest neighbour matching. The accuracy of a match is verified using three tests. The first test is the second nearest neighbour ratio test described in [39]. The second test checks that the matched feature belongs to the correct object. The last test uses the epipolar constraints of the two images containing the matched features and an auxiliary view.

Unfortunately the camera calibration information is not provided with the image data. The calibration is also not consistent throughout the database due to problems encountered during the acquisition. These problems make using this database and ensuring the quality of the results excessively difficult. The dataset is available at `http://www.vision.caltech.edu/pmoreels/Datasets/TurntableObjects/`.

3.4 Local Feature Extractors, Descriptors and Matching in the Literature

3.4.1 Feature Extractors

A very large number of local feature extractors have been proposed for various applications. A detailed review has been published in [54]. Two performance evaluations have been presented in [45] and [47]. Among the array of available feature extractors, only those that are affine covariant are sufficiently robust to aid in the calibration of widely separated cameras. The following affine covariant feature extractors are reviewed in this section. The Harris Affine and Hessian Affine extractors [43, 45] are part of a group of extractors referred to as saliency map-based feature extractors. This class of extractors is studied extensively in this thesis. The Maximally Stable Extremal Regions (MSER) extractor [40] achieved the best results in many of the performance tests and is used in some experiments in this thesis. Other affine covariant feature extractors include Edge Based Regions (EBR) and Intensity Based Regions (IBR) [55, 56, 58], Salient Regions [24, 25] and Stable Affine Frames (SAF) [48].

The work in this thesis is based on three extractors – MSER, Harris Affine and Hessian Affine. These three extractors achieve superior results in the published evaluations and make use of some of the most thoroughly researched and tested techniques.

Saliency Map-based Extractors

This class of extractors operates by computing a saliency map from an image and selecting the local maxima or minima of this map as points of interest. The saliency map is a function of image partial derivatives and indicates regions of high information content. Various saliency operators have been proposed, including the Harris & Stephens operator [21], the Determinant of Hessian operator [6], and the Laplacian of Gaussian and Difference of Gaussians [39] operators. Features extracted using a saliency map are not directly affine covariant, but are only isometrically covariant. Affine covariance is achieved by estimating the scale and shape of the features through a process called affine adaptation.

Certain saliency map-based extractors have been shown to perform relatively well (the Hessian and Harris Affine extractors [43, 45] in particular). Although the MSER extractor (discussed below) requires less processing time per image and has been shown to be more repeatable in some tests, the saliency map-based extractors produce larger numbers of features and are superior in scenes that do not contain many planar features [47]. Saliency map-based extractors and affine adaptation are discussed in detail in Chapters 4 through 6. Much of the research reported in this thesis makes contributions in the field of saliency map-based feature extractors.

Maximally Stable Extremal Regions

The operation of the MSER extractor [40] is analogous to sequentially applying a range of thresholds to an image and keeping track of the connected sets of pixels above and below the threshold. These connected sets of pixels are called extremal regions because either all the pixels in a region have higher value than the region border, or they all have lower value. An extremal region is considered maximally stable at a given threshold if the change in region size over a local range of thresholds is minimal.

The output of the MSER extractor is the set of pixel coordinates for each region. This may be converted to the feature transformation representation (see Section 3.2.2) by computing the transformation parameters from the first and second order moments of the pixel locations. The extractor algorithm is very similar to the watershed algorithm and may be implemented in a computationally efficient manner [10, 15, 28].

MSER attained the highest scores in many of the tests performed in the evaluations [45, 47]. It is most effective when the scene contains many planar regions.

When a scene does not contain many planar regions, the performance of the MSER extractor falls below that of the saliency map-based methods. The most significant shortcoming of MSER is that it produces significantly fewer correspondences than other methods.

Edge Based Regions Extractor

The EBR extractor [55, 56, 58] uses the Harris & Stephens operator [21] to select corner points. The Canney edge detector [9] is then used to extract nearby edges in the image. The Harris corner is labelled $\mathbf{p}$ and a point on edge i leading away from $\mathbf{p}$ is labelled $\mathbf{p}_i(s_i)$, where s_i is an arbitrary parameter of the edge. A relative affine invariant parameter is defined for each point as,

$$l_i = \int \text{abs}\left(\begin{vmatrix} \frac{\partial \mathbf{p}_i}{\partial s_i} & \mathbf{p} - \mathbf{p}_i(s_i) \end{vmatrix} \right) ds_i,$$

where abs() is the absolute value. The ratio of two of these relative invariants, $\frac{l_i}{l_j}$, is an absolute affine invariant. By requiring that $l = l_1 = l_2$ for two points, $\mathbf{p}_1$ and $\mathbf{p}_2$, and defining a point $\mathbf{q}$ that completes the parallelogram defined by the points $\mathbf{p}, \mathbf{p}_1, \mathbf{q}, \mathbf{p}_2$, a region, $\Omega(l)$, is defined that is a function of the parameter l.

Regions are selected from $\Omega(l)$ where both of the following photometric quantities go through an extreme value over l,

$$\begin{aligned} P_1 &= \text{abs}\left(\frac{\begin{vmatrix} \mathbf{p}_1 - \mathbf{p}_g & \mathbf{p}_2 - \mathbf{p}_g \end{vmatrix}}{\begin{vmatrix} \mathbf{p} - \mathbf{p}_1 & \mathbf{p} - \mathbf{p}_2 \end{vmatrix}} \right) L, \\ P_2 &= \text{abs}\left(\frac{\begin{vmatrix} \mathbf{p} - \mathbf{p}_g & \mathbf{q} - \mathbf{p}_g \end{vmatrix}}{\begin{vmatrix} \mathbf{p} - \mathbf{p}_1 & \mathbf{p} - \mathbf{p}_2 \end{vmatrix}} \right) L, \end{aligned}$$

where,

$$\begin{aligned} L &= \tfrac{M_1}{\sqrt{M_0 M_2 - M_1^2}}, \\ M_n &= \textstyle\int_\Omega I^n(x,y)dxdy, \\ \mathbf{p}_g &= \tfrac{1}{M_0}\begin{bmatrix} \int_\Omega I(x,y)xdxdy \\ \int_\Omega I(x,y)ydxdy \end{bmatrix}. \end{aligned}$$

In the case where both edges are straight, $l = 0$ and the above method cannot be used. In these cases, the constraint $l = l_1 = l_2$ is not used and Ω is instead defined in terms of $s1$ and $s2$. Values for $s1$ and $s2$ are chosen where the valleys of P_1 and P_2 intersect. The minima of P_1 and P_2 are not used, since they are not well defined in this case.

The EBR extractor performs relatively poorly in the majority of tests in [45]. It achieved at best similar results to Harris Affine and Hessian Affine, but scored significantly lower than these extractors in many of the tests. It also requires two orders of magnitude more computation time per image.

Intensity Based Region Extractor

The IBR extractor [55, 56, 58] selects extrema of the image intensity function as region centre points. It then evaluates the following function along rays extending outward from the selected region centre:

$$f(r) = \frac{\text{abs}(I(r) - I_0)}{\max\left(\frac{\int_0^r \text{abs}(I(r)-I_0)dr}{r}, \epsilon\right)},$$

where r is the radius from the centre point, I_0 is the image intensity at the centre point, and ϵ is a small number used to prevent dividing by zero. The region boundary is marked by finding the first local maximum along $f(r)$ for a set of rays emanating from the centre point. An ellipse is fitted to these points by computing their second order moments. Finally, the size of the ellipse is multiplied by two.

The main shortcoming of IBR is that it produces very few features compared to other extractors. It obtains good repeatability in the change of viewpoint test with a structured scene, but performs poorly in all other tests [45, 47].

Salient Regions

The Salient Regions extractor [24, 25] evaluates an entropy function over all possible elliptical regions in the image (a five parameter space corresponding to the ellipse parameters). For a given ellipse, the entropy is computed as,

$$\mathcal{H} = -\sum_{I} p\left(I\right) \log p\left(I\right),$$

where $p\left(I\right)$ is the probability density function of the image intensities in the elliptical region. Where the entropy reaches maxima over scale, the entropy is multiplied by the derivative of the pdf over scale,

$$\mathcal{W} = \frac{s^2}{2s-1} \sum_{I} \left| \frac{\partial p\left(I\right)}{\partial s} \right|,$$

to compute the saliency measure, $\mathcal{Y} = \mathcal{W}\mathcal{H}$ (s being the scale parameter).

This extractor is extremely time consuming – it requires on the order of 30 minutes per image. This is due to the exhaustive evaluation of entropy over a five parameter affine scale space. It also achieves the lowest scores of all the tested extractors in the majority of tests.

Stable Affine Frames Extractor

The SAF extractor [48] constructs affine frames on intensity isophotes and selects frames that are sufficiently stable. Affine frames are formed by finding sets of three points on an isophote – two points defining a bitangent and the furthest point from the bitangent along the isophote between the bitangent points. The bitangents are found using the method in [8]. Stable frames are selected by grouping frames in adjacent intensities using a similarity measure. A group of similar frames is said to be stable if the group contains a sufficient number of similar frames.

The evaluation of the SAF extractor in [48] presents a limited set of results from the test presented in [45] (SAF was developed subsequent to the publication of [45, 47] and was not included in those evaluations). From the limited results it appears that the repeatability of the SAF extractor is on par with MSER, Hessian Affine and Harris Affine, while it consistently produces more correspondences. No matching score results were published. This extractor appears to be an interesting candidate for wide baseline matching, however, due to time constraints it could not be implemented and properly evaluated.

3.4.2 Descriptors

Once features have been extracted from multiple views, they must be compared and matched across views. Comparing their support region images or normalised support region images is both computationally expensive and ineffective. It is computationally expensive because a significant number of pixels for each feature from one image must be compared to the pixels of every feature from a second image. The complexity is therefore a product of the number of sample pixels, and the number of features in both images. It is ineffective because features do not provide perfectly aligned images, even when normalised. This is due to the fact that features are extracted independently from each image and are not aligned directly. Feature descriptors offer a better solution to the problem, because they provide a more compact representation of features, require fewer comparison operations, and they introduce a limited degree of robustness to slight variations in feature appearance. A vector representation also allows the set of features to be stored in a structure that is more efficiently searched, such as a k-D tree.

Evaluations of feature descriptors have been published in [42, 44, 47]. In these evaluations the descriptors based on the Scale Invariant Feature Transform (SIFT) [38, 39] perform the best. The SIFT descriptor is used throughout this thesis. Some local feature descriptors are reviewed below.

Histogram-based descriptors

Descriptors that make use of histogram techniques have been very successful. They are effective because they capture the shape of local features very well and encode this shape in a high dimensional vector. The Scale Invariant Feature Transform (SIFT) is the most significant among these and is the inspiration for the other methods.

SIFT, as proposed in [38, 39], consists of a feature extractor and descriptor. This discussion will focus on the descriptor only. SIFT captures the distribution of gradients in a local image region by constructing a 3D histogram of gradients. The histogram dimensions consist of the x and y position in the normalised image and the orientation of the local gradient. A typical configuration divides the spatial dimensions into 4 bins and the gradient orientation into 8 bins. This gives a total of 128 bins, which are used directly as a 128-dimensional descriptor vector.

The intensity gradient magnitude and orientation is computed at each pixel in the normalised image. The gradient is weighted by a Gaussian window centred at the normalised image centre, in order to give the central regions greater weighting. Each gradient is distributed to the bins that are nearest to its position and orientation using a trilinear weighting method, to avoid boundary effects. The descriptor vector is composed from the elements of the histogram. The vector is normalised by normalising its magnitude to 1, applying a threshold of 0.2 to all elements and, finally, normalising the magnitude to 1 again.

Shape Context [7] is similar to SIFT, except that it compiles a histogram of edge locations (instead of all gradients) using a log-polar distribution of bins. A modified version of Shape Context is presented in [44], where the edge points are also binned according to the edge orientations and weighted according to edge gradient magnitude, making it even more similar to SIFT. In most tests, Shape Context performs similarly to SIFT, but it performs significantly worse in viewpoint change tests. Its main shortcoming is the limited robustness of the edge extraction process – the same edge pixels are not always selected for similar features.

The Gradient Location-Orientation Histogram (GLOH) descriptor is an extension to SIFT proposed in [44]. It computes a histogram in a similar manner to SIFT, but using a different spatial distribution of bins and a larger number of bins. The resulting descriptor vector is then projected to a 128 dimensional vector. The transformation used for the projection is computed using principal component analysis on a large set of training descriptors. GLOH performed better than SIFT in some tests, but was not significantly better than SIFT on average.

Spin Images [29, 30] builds a histogram according to the radius from the feature centre and intensity value. Constructing the histogram in this way results in a rotation invariant descriptor. Ten bins are used for both radius and intensity, producing a vector with 100 dimensions. The concept was inspired by a method used to describe and match 3D surface meshes [23]. Spin Images performs relatively poorly in all tests, when compared to other descriptors. A similar variation of SIFT is also presented in [30], called Rotation Invariant Feature Transform (RIFT). It uses radial bins instead of a square grid of bins, thereby reducing the sensitivity to rotation. The RIFT descriptor was not tested against the other descriptors in an equivalent test, however, results from [30] indicate that its performance is significantly reduced compared to SIFT.

Other descriptors

PCA-SIFT was introduced in [26]. The SIFT feature extractor was used, but the descriptor is not related to SIFT at all. A linear transformation is used to project the image of local gradients to a low dimensional (20 dimensions in [26]) vector space. The transformation is computed by means of principal component analysis on a large set of normalised gradient images found by extracting features from a set of training images. PCA-SIFT performed the best among the methods that do not use histograms, but its performance is inferior to the SIFT-based descriptors.

Steerable filters [20, 27] measure the response of a set of filters to the normalised image. A steerable filter is a linear combination of Gaussian derivative basis functions. Complex filters [51] operate in a similar manner. In this case the filters used are derived from the family,

$$f_{mn}\left(x,y\right) = \left(x+iy\right)^{m}\left(x-iy\right)^{n} g\left(x,y,\sigma\right),$$

with σ fixed proportional to the normalised image size. These filter-based approaches performed relatively poorly in all tests. The number of descriptor dimensions is determined by the number of filters applied and is therefore typically relatively low. Higher order derivatives cannot be used to increase the number of dimensions, because they are too sensitive to minor variations and detract from the robustness of the descriptor.

Moment invariants have been proposed in [46, 57]. Generalised colour moments of order $p+q$ and degree $a+b+c$ are defined as,

$$M_{pq}^{abc} = \iint_{\Omega} x^p y^q R^a(x,y)\, G^b(x,y)\, B^c(x,y)\, dxdy,$$

where R, G and B are the images of the red, green and blue channels and Ω is the computation window. Moment invariants produce descriptors with few dimensions. Higher order moments are sensitive to small geometric variations and are no more robust than correlation methods.

Speed Up Robust Features (SURF) [5] is a feature extractor and descriptor intended to require little processing time. The descriptor is computed as follows. The normalised image region is divided into a regular 4×4 grid. In each grid sub-region the Haar wavelets in the horisontal and vertical directions (d_x and d_y) are sampled from a 5×5 grid of sample points. Four quantities are computed from the Haar wavelet samples,

$$\begin{aligned} v_1 &= \sum d_x, \\ v_2 &= \sum |d_x|, \\ v_3 &= \sum d_y, \\ v_4 &= \sum |d_y|. \end{aligned}$$

These values are collected from all of the 4×4 grid of subregions to form a descriptor vector with 64 dimensions. The descriptor vector is then normalised to have unit length. An alternative formulation is also proposed that has 128 dimensions and is called SURF-128. For each subregion, the following quantities are computed,

$$\begin{aligned} v_1 &= \sum_{\forall d_x<0} d_x, & v_2 &= \sum_{\forall d_x<0} |d_x|\,, \\ v_3 &= \sum_{\forall d_x>0} d_x, & v_4 &= \sum_{\forall d_x>0} |d_x|\,, \\ v_5 &= \sum_{\forall d_y<0} d_y, & v_6 &= \sum_{\forall d_y<0} |d_y|\,, \\ v_7 &= \sum_{\forall d_y>0} d_y, & v_8 &= \sum_{\forall d_y>0} |d_y|\,. \end{aligned}$$

From the evaluation given in [5] it appears that the performance of SURF is similar to that of the SIFT-based descriptors while the performance of SURF-128 is superior. The evaluation in [5] is unfortunately very limited (only one test case is used). More thorough evaluation would be required before making any conclusions regarding the performance of the SURF descriptor.

3.4.3 Matching

Several different techniques have been proposed for finding matches between two sets of features. In general, matching involves evaluating the distance between features. Euclidean distance in the descriptor vector space is the most common method used.

Threshold-based matching simply applies a threshold to the distance between every pair of features. All feature pairs with a distance below the threshold are matched. This results in a set of one-to-many matches. That is, each feature in one image may be matched to several features in a second image.

In nearest neighbour matching, each feature in one image is matched to the nearest feature in the second image if the distance between these features is below a threshold. Each feature may be matched to one feature only.

Nearest neighbour distance ratio matching also involves finding the nearest neighbour matches, but applies the threshold differently. A nearest neighbour match is rejected if the ratio between the distance to the nearest neighbour and the distance to the second nearest neighbour is above a threshold. If the distance from a feature to its nearest neighbour is D_{n1} and the distance to the second neighbour is D_{n2}, then the threshold, t, is applied as follows: $D_{n1}/D_{n2} < t$. This method was first described in [39]. It effectively avoids selecting ambiguous matches, thereby greatly reducing the number of incorrect matches, when compared to nearest neighbour matching. This method is used for matching in the work presented in this thesis.

More elaborate approaches to feature matching have been proposed. In [2] the matching problem is treated as a classification problem. In [41] a method is proposed for projecting descriptor vectors to a lower dimensional space so that the nearest neighbour search method becomes optimal (under the assumption that descriptor variation is Gaussian distributed). The reduction in descriptor dimensionality aids in reducing the computational cost of matching. Bayesian and maximum a posteriori, approaches are presented in [13].

3.5 Chapter Summary

In this chapter, the problem of aligning images taken from widely separated viewpoints is formulated in terms of the wide baseline matching problem. The local image feature-based approach of feature extraction, feature description and matching, is reviewed. Although the local feature-based methods in general can be used in a wide variety of problems, the review in this chapter and the research in the following chapters is focused on the techniques relevant to the wide baseline case. Features that are robust to affine deformation, or affine covariant features, are required in this scenario.

From the review in this chapter, specific techniques are selected for use in further research. Among the affine covariant local feature extractors, the saliency map-based methods and the MSER extractor yield the best results. These extractors are used and developed further in the following chapters. The evaluation method presented in Section 3.3.1 is used to evaluate the results of new methods for saliency map-based feature extraction. An efficiency metric was added to the existing set of tests to evaluate the relative computational efficiency of different feature extractors. The SIFT descriptor and nearest neighbour ratio matching is used to established correspondences.

The next chapter reviews saliency map-based feature extraction and adaptation in greater detail.

Chapter 4

Salient Features

A number of the more successful affine covariant feature extractors make use of a saliency map to detect features, and adaptation processes to extract the shape and size of each feature. A saliency map is computed using an image operator that is a function of image partial derivatives. The saliency operator highlights local features that are suitable for matching across views, but does not provide a viewpoint covariant feature directly.

The output of saliency operators is dependant on the scale or resolution of an image. Section 4.1 introduces the concept of scale space and explores the properties of the Gaussian scale space. Section 4.2 discusses the computation of the saliency map, existing saliency operators and properties of the saliency map.

The saliency map provides a means of determining the feature coordinates, t_x and t_y. Each of the remaining parameters (scale k, shape q, ϕ, and orientation θ) is determined using a separate adaptation process. Section 4.3 reviews techniques for selecting scale, shape and orientation for features detected using a saliency map.

In the literature, feature extractors are usually discussed in terms of a specific solution to all of the above problems, while adhering to the same overarching design model. A generalised saliency map-based feature extractor algorithm is presented in Section 4.4. It is used in the rest of this thesis to directly compare alternative feature parameter estimation methods within the same framework.

Section 4.5 discusses how to implement some of the operators mentioned in this chapter in an efficient manner. The chapter is concluded in Section 4.6

4.1 Scale Space

Scale is a very important concept in computer vision. The description of an object depends greatly on the size of the object and of the scale at which it is described. The scale of objects is not preserved in an image and is affected by factors such as focal length, resolution and the distance between camera and subject. Describing an object from its image therefore requires finding an appropriate scale at which to describe it, or perhaps describing it over a range of scales. The saliency operators described in later sections are sensitive to a change in scale – images of the same scene taken at different resolutions or with different camera zoom settings result in saliency maps where the extrema are not in comparable locations.

The concept of scale space has been formalised in order to analyse images with objects of different scales. Changing the zoom level of a camera or moving the camera towards or away from a subject effectively alters the relative resolution with which the projection of the subject is sampled. Scale space models the process of creating an image with varying resolution, facilitating multi-resolution or multi-scale analysis. An image scale space is a representation of an image that provides access to different image scales by means of a scale parameter. It is generated by progressively removing details from the image, so that a higher scale level contains only larger scale detail and all smaller scale detail is removed. The scale parameter corresponds to the parameter of the process by which details are removed. A low pass filter is a natural choice for the process described above.

Scale space theory is discussed at length in [33, 53]. This section describes the 2D Gaussian scale space and its properties.

4.1.1 The Isotropic Gaussian Scale Space

It is desirable for the filter function chosen for producing a scale space to possess the following properties:

- Rotation and shift invariance, to ensure the scale space is not affected by image transformations that do not affect the image scale.
- Strictly decreasing magnitude around the filter coordinate origin.
- Consistent behaviour over all scales.
- Differentiable and integrable, since it is desirable to work with the derivatives of the scale space.
- The number of local maxima and minima of an image and its derivatives strictly decrease as scale increases. This ensures that the level of detail strictly decreases as scale increases.

The Gaussian function possesses all of these properties. In [1], it is argued that the Gaussian function is the only function that ensures that the number of local extrema strictly decrease as scale increases. Artefacts of the imaging process, such as blurring due to limited lens resolution or incorrect focus, may also be

Figure 4.1: The 2D Gaussian function. Truncated, scaled and contrast enhanced for presentation. Black corresponds to zero and brighter intensity represents higher value.

modelled using the Gaussian function. Although other definitions of scale space do exist, the Gaussian scale space (defined below) is the preferred definition of scale space for use in robust local feature extraction.

The isotropic Gaussian operator is defined in terms of inhomogeneous coordinates $\mathbf{x} = (x, y)^\top$ and variance σ^2 as,

$$g(\mathbf{x}, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-\mathbf{x}^\top \mathbf{x}}{2\sigma^2}} = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2 - y^2}{2\sigma^2}}. \tag{4.1}$$

Figure 4.1 shows a Gaussian function. The isotropic Gaussian scale space of image $I(\mathbf{x})$ is defined simply as the convolution of the image with the Gaussian kernel $g(\mathbf{x}, \sigma)$,

$$I(\mathbf{x}, \sigma) = g(\mathbf{x}, \sigma) * I(\mathbf{x}). \tag{4.2}$$

This space then has coordinates (x, y, σ).

A discrete scale space may be computed by sampling the space at discrete intervals of σ. Such a representation is often called the scale space pyramid and the individual images that make up the pyramid are referred to as layers of the pyramid. Because higher scale images contain less information and do not contain high resolution information, it is possible to sample these layers at a lower spatial density. This reduces the amount of memory and processing required.

A common approach (used for example in [39]) is to sample the scale space at regular intervals of $\log_2(\sigma)$. The same spatial sampling density is used over a scale octave. This ensures that the pixels of each level in an octave are consistently aligned, which simplifies multi-scale image operations. Each successive octave is down-sampled by a factor of two.

4.1.2 Scale Space Derivatives

Saliency operators are mostly based on image derivatives. It is therefore important to understand the properties of image derivatives in scale space. This section discusses the properties of scale space derivatives and Section 4.1.3 discusses how the extrema of derivative functions are affected by scale.

The following notation will be used to express scale space derivative images more concisely,

$$I_j(\mathbf{x}) = \frac{\partial}{\partial j} I(\mathbf{x}).$$

Image and Gaussian Derivatives are Interchangeable

The scale space derivative expression may be stated in any of the following forms:

$$\frac{\partial}{\partial i} I(\mathbf{x}, \sigma) = \frac{\partial g(\mathbf{x}, \sigma)}{\partial i} * I(\mathbf{x}) = g(\mathbf{x}, \sigma) * \frac{\partial I(\mathbf{x})}{\partial i}, \tag{4.3}$$

where i is any of the spatial dimensions. The derivative of a scale space image, $I(\mathbf{x}, \sigma)$, produced by convolution with a Gaussian, is equivalent to convolving the derivative of the Gaussian with the image and equivalent to convolving the Gaussian with the derivative of the image. More generally, since the Gaussian is infinitely differentiable, for any order and sequence of partial derivatives (including derivatives with respect to scale),

$$\frac{\partial^n}{\prod_{j=1}^{n} \partial i_j} I(\mathbf{x}, \sigma) = \frac{\partial^n g(\mathbf{x}, \sigma)}{\prod_{j=1}^{n} \partial i_j} * I(\mathbf{x}).$$

The properties of scale space derivatives are therefore determined directly by the properties of the Gaussian function. This has implications for the implementation of the saliency and scale space image operators (see Section 4.5).

Non-stationary extrema

Extrema of scale space derivatives do not necessarily remain spatially stationary over a change in scale (except perhaps in the centre of rotationally symmetric image structures). The exact behaviour of extrema over scales is explored in Section 4.1.3.

Decreasing amplitude over scale

Consider an image, $I(\mathbf{x})$, that contains a single step edge that runs perpendicular to dimension i. The maximum value of the partial derivative of the image with respect to dimension i decreases as the scale increases. If $I_i(\mathbf{x}, \sigma_1)$ and $I_i(\mathbf{x}, \sigma_2)$ are the partial derivatives along dimension i of the image at scale levels $\sigma_1 < \sigma_2$, then,

$$\arg\max_{\mathbf{x}} |I_i(\mathbf{x}, \sigma_2)| < \arg\max_{\mathbf{x}} |I_i(\mathbf{x}, \sigma_1)|.$$

This property is related to the fact that a change in sampling resolution will result in a change in derivative amplitude, unless the resolution is accounted for. If the derivative of a function sampled with period p, $f(px)$, $x \in (Z)$, is to be computed correctly, then the sampling period must be accounted for by adjusting the unit length,

$$\frac{\Delta f(x)}{\Delta x} = \frac{\Delta f(px)}{p\Delta x}.$$

In the case of the Gaussian function, $g(\mathbf{x}, k\sigma) = g(k^{-1}\mathbf{x}, \sigma)$, which means that changing scale is analogous to changing sampling resolution, and,

$$\frac{\partial g(\mathbf{x}, \sigma)}{\partial i} = k\frac{\partial g(\mathbf{x}, k\sigma)}{\partial i}.$$

The magnitude of derivatives is not only affected by scale according to the above property, but also due to the image structure. It is desirable to extract information regarding image structure from the change in derivative magnitude over scale. It is therefore necessary to normalise scale space derivative images according to scale and the order of derivative as follows,

$$I_{\mathrm{n}i^m}(\mathbf{x}, \sigma) = \sigma^m \frac{\partial^m}{\partial i^m} I(\mathbf{x}, \sigma).$$

4.1.3 Extrema of Scale Space Derivatives

Multi-scale analysis is motivated by the fact that the positions of extrema of scale space derivatives are not scale covariant, but vary over a change in scale. Figure 4.2 shows four levels of a scale space pyramid, as well as the Determinant of Hessian operator (see Section 4.2.1) computed on each of the levels of the pyramid. The locations of the maxima of the Determinant of Hessian map vary significantly over scale. Although different saliency operators behave differently, they are all based on scale space derivatives and share common characteristics. These characteristics are explored in detail below. In this section, the extrema (over the spatial dimensions) of saliency operators will be referred to as extrema.

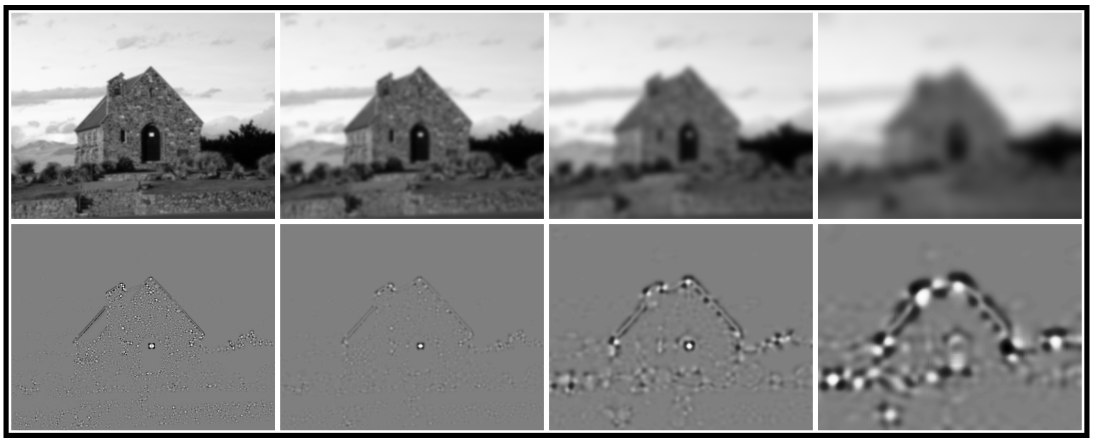


Figure 4.2: Deterimanant of Hessian computed at various scales.

Continuous Variation over Scale

The spatial positions of extrema vary continuously over scale. The velocity of extrema with respect to scale is determined by the image structure, and is explained in the properties listed below.

Motion Away from Edges

For all the saliency functions described in this chapter, extrema are found near two or more (non-parallel) straight edges or near curved edges. As scale increases, extrema move away from the edges of origin so that the displacement component perpendicular to each edge is proportional to the change in scale. Extrema move from a narrower region to a wider region over increasing scale. When an extreme point reaches a location in a bounded region where it is at a maximum distance from all the enclosing edges, the extreme point becomes stationary, since it can no longer move further from all of the dominant edges surrounding it (until structures from outside the bounded region begin to have a significant affect). Figure 4.3 shows an example where the Laplacian of Gaussian of various scales is convolved with a triangle.

Consider a corner with angle θ formed by the intersection of two step edges of equal step size as an example. The extreme point will be found on the line bisecting the corner and will travel away from the corner intersection. Over a change in scale $\Delta\sigma$, the extreme point will be displaced by distance d along the bisecting line of the corner, which may be expressed as $d \propto \Delta\sigma / \sin(\theta/2)$.

In a triangle, three extrema will originate in the corners and move towards the centre of the incircle of the triangle. The edge opposite each corner will have no effect on the extreme point formed by the corner at low scale, due to the relatively large distance to the edge. As scale increases, this edge will at first attract the extreme point, so that it accelerates until it converges with the other two extrema from the opposite corners. Once the extrema converge to one point, this extreme point will be effected by all three edges equally and will remain stationary as scale increases. The velocity of extrema is therefore only strictly linear when no more than two straight edges are present.

Converging Extrema

Similar extrema can converge as scale increases. Maxima can converge and minima can converge, but a maximum and a minimum cannot converge to the same point. A single extreme point cannot diverge into multiple extrema as scale increases, due to the characteristics of the scale space. Extrema accelerate towards each other as they approach convergence. Extrema related to different structures are, by definition, separated by edges and as such will be located a significant distance apart, may be of different sign and will propagate in diverging directions as scale increases (at least until the separating structure is no longer dominant).

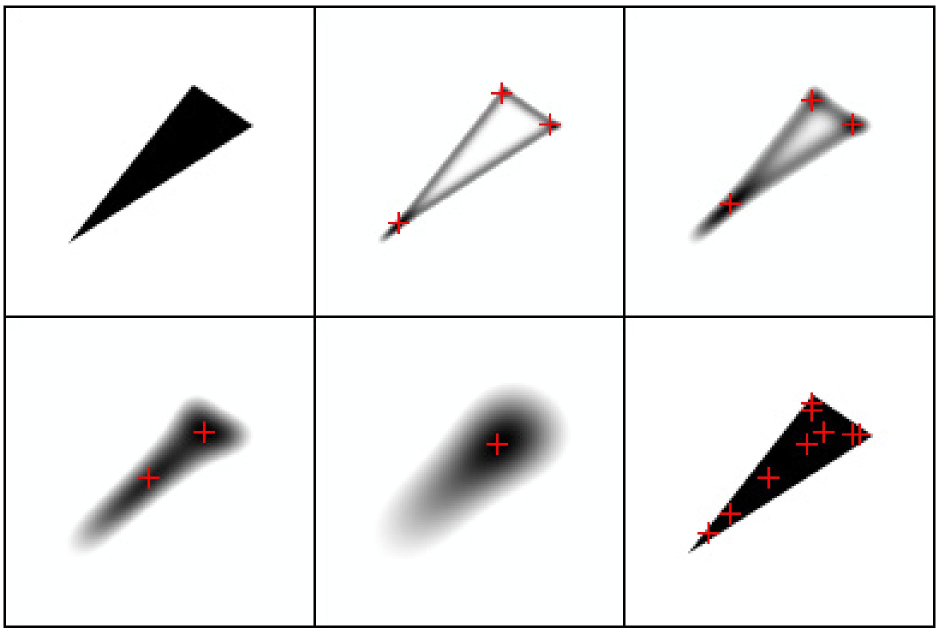


Figure 4.3: The top left frame shows a triangle input image. The next four frames show the image convolved with the Laplacian of Gaussian function at scales 2, 4, 8 and 16. The maxima are indicated by red crosses. The last frame shows the positions of the maxima overlaid on the original image.

Creation and Annihilation

Extrema may both be created and annihilated as scale increases. Consider a perfectly circular image structure convolved with a Laplacian of Gaussian kernel (introduced in the next section). If the scale of the Laplacian is sufficiently low, then it will result in a circular ridge and valley on either side of the circle edge, but no locally extreme point. As the scale increases, the ridge and valley will move away from the edge. An extreme point will be created at the centre of the circular shape at the scale where the ridge or valley on the inside of the circle converges on the centre. In the presence of large, distant structures around the circle, the extreme point will, at a sufficiently high scale, become insignificant compared to the response of the Laplacian to other structures, and will be annihilated.

It should be noted that image intensity extrema cannot be created in the scale space as scale increases. Extrema can only be created in the derivatives of the scale space.

4.2 The Saliency Map

The purpose of the saliency map is to give a measure of the local information content in an image, in order to aid in locating regions suitable for matching across views. Image structures that are spatially well defined result in an interesting response from the saliency function, usually in the form of a locally extreme value. Features are selected at the maxima or minima (depending on the saliency function used) of the saliency map. A threshold is often applied to suppress noise and responses to weak structures.

4.2.1 Saliency Functions

A large number of saliency operators have been proposed for the task of image feature detection. This section defines the operators that have been used for the task of wide baseline matching with most success. Section 4.5 reviews techniques for implementing these operators.

Laplacian of Gaussian and Difference of Gaussians

The Laplacian of image $I(\mathbf{x})$ is defined as,

$$\mathcal{L}(I(\mathbf{x})) = \nabla^2 I(\mathbf{x}) = \frac{\partial^2 I(\mathbf{x})}{\partial x^2} + \frac{\partial^2 I(\mathbf{x})}{\partial y^2}. \tag{4.4}$$

The Laplacian of Gaussian (LoG) function is defined as,

$$\mathcal{L}_g(\mathbf{x}, \sigma) = \nabla^2 g(\mathbf{x}, \sigma) = \frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^6} e^{\frac{-x^2 - y^2}{2\sigma^2}}, \tag{4.5}$$

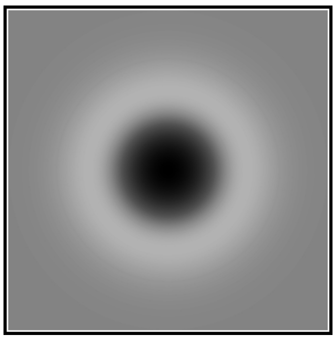


Figure 4.4: The 2D Laplacian of Gaussian function, truncated, scaled and contrast enhanced for presentation. 50% grey value (near image corners) corresponds to a value of zero.

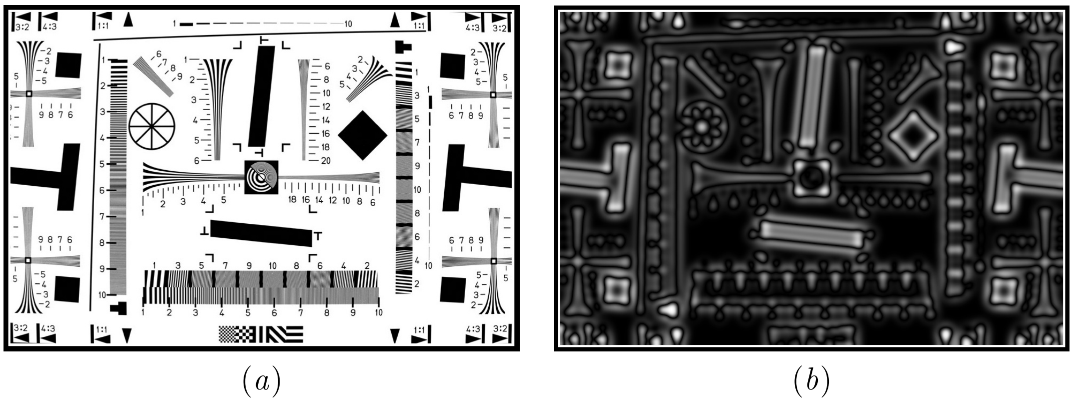


Figure 4.5: (a) A test pattern. (b) The magnitude of the test pattern after convolving with a LoG kernel. Black corresponds to zero and brighter intensity represents higher value.

and is shown in Figure 4.4. Figure 4.5 shows the result of convolving an image with this function and taking the absolute value at each pixel.

This operator produces a large magnitude response in the presence of steep gradients or edges in the image. Curved edges elicit an even larger response, especially curves with radius σ. Corner and blob structures result in local extrema in the output. The LoG operator is therefore useful for corner and blob detection. It also produces spurious local extrema along straight edges due to noise or slight curvature changes. This results in a significant proportion of local extrema associated with noise or structures that cannot be located reliably across views.

The first order derivative of the Gaussian along the scale dimension is equivalent to the scale-normalised Laplacian of Gaussian,

$$\frac{\partial}{\partial \sigma} g\left(\mathbf{x}, \sigma\right) = \sigma^2 \nabla^2 g\left(\mathbf{x}, \sigma\right).$$

The Difference of Gaussians operator is an approximation of the the derivative over scales (and the Laplacian) and is computed by subtracting two Gaussians as follows,

$$\mathcal{D}_g\left(\sigma_1, \sigma_2\right) * I\left(\mathbf{x}\right) = \left(g\left(\mathbf{x}, \sigma_2\right) - g\left(\mathbf{x}, \sigma_1\right)\right) * I\left(\mathbf{x}\right) = I\left(\mathbf{x}, \sigma_2\right) - I\left(\mathbf{x}, \sigma_1\right),$$

where $\sigma_2 > \sigma 1$. It is a close approximation to the scale-normalised Laplacian of Gaussian and, given an efficiently implemented Gaussian filter, is considerably more efficient to compute than convolving an image with the Laplacian kernel.

A more general form of the Difference of Gaussians, the Difference of Low Pass (DoLP) transformation was introduced in [12] in a framework for describing shapes in a multi-scale representation. Examples of previous works that make use of the LoG or DoG operator for feature detection include [34, 38, 39]. It is also used for scale selection (see Section 4.3.1).

Determinant of Hessian

The Hessian is the matrix of second order partial derivatives of a function. The Hessian of an image convolved with a Gaussian is defined as,

$$\frac{\partial^2 I\left(\mathbf{x}, \sigma\right)}{\partial \mathbf{x}^2} = \begin{bmatrix} \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial x \partial y} \\ \frac{\partial^2}{\partial x \partial y} & \frac{\partial^2}{\partial y^2} \end{bmatrix} I\left(\mathbf{x}, \sigma\right) = \begin{bmatrix} I_{x^2}\left(\mathbf{x}, \sigma\right) & I_{xy}\left(\mathbf{x}, \sigma\right) \\ I_{xy}\left(\mathbf{x}, \sigma\right) & I_{y^2}\left(\mathbf{x}, \sigma\right) \end{bmatrix}. \quad (4.6)$$

Its eigenvalues represent the second order gradients in the directions of its eigenvectors. The determinant of this matrix is large when both eigenvalues are large, and gives an indication of regions where there are strong gradients in more than one direction. In the proximity of a straight edge or parallel edges, only one eigenvalue is large and the determinant has small magnitude. The determinant of the Hessian matrix is therefore useful for detecting corners and blobs, while unstable structures and noise along nearly straight edges are suppressed.

Figure 4.6 illustrates the process of applying the determinant of Hessian operator. Observe that, in comparison to the Laplacian of Gaussian operator in Figure 4.5, the straight edges in the image do not produce a large response. Only spatially well defined structures such as blobs and corners produce significant local maxima.

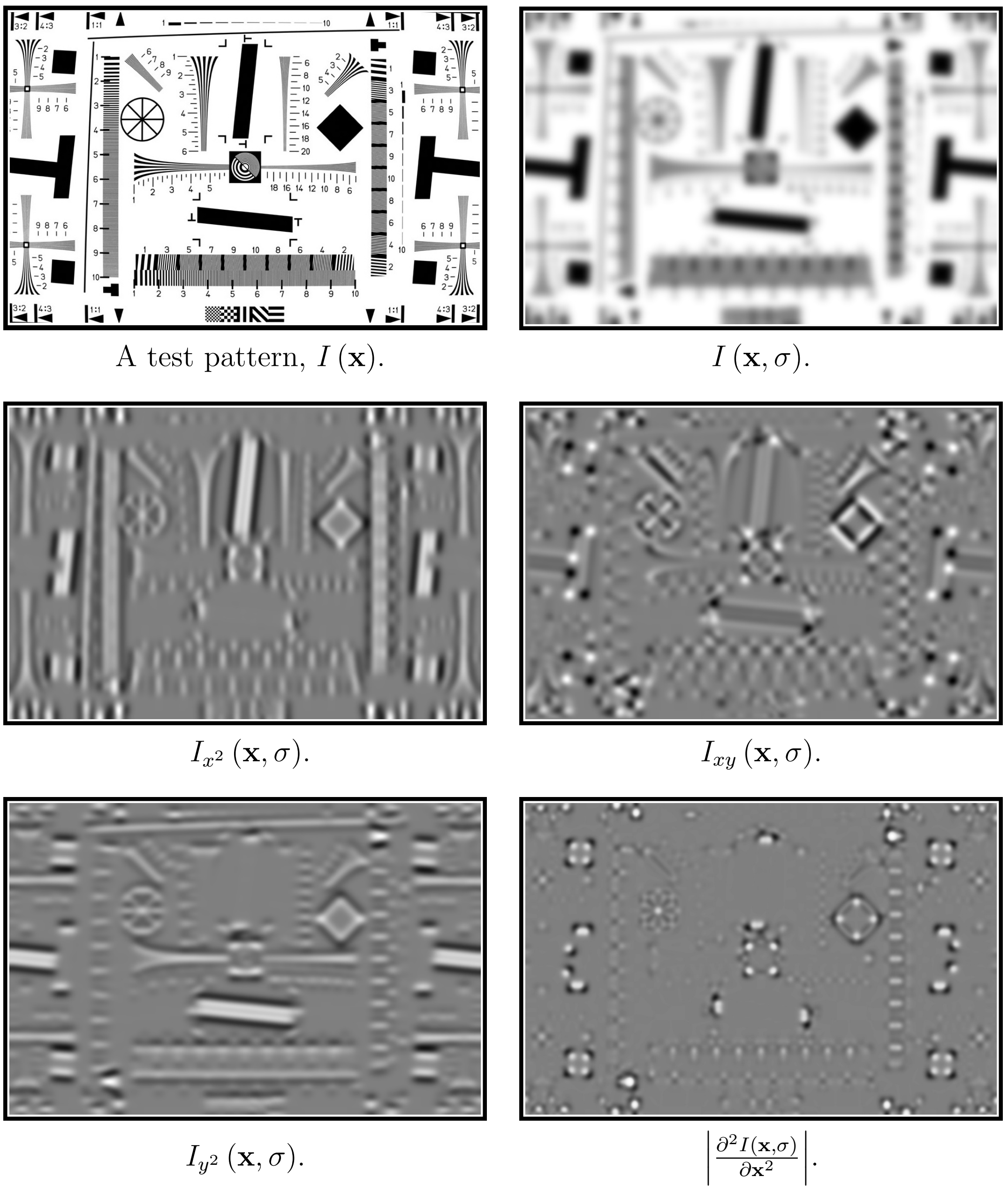
The Determinant of Hessian operator was first introduced in [6] and is one of the earliest operators introduced for corner selection. It has recently been used to create some of the most successful feature extractors [43, 45].

Harris

The Harris operator [21] involves applying an operator commonly referred to as the second moment matrix and then computing a saliency measure from this matrix. The second moment matrix is defined as,

$$\mu\left(\mathbf{x},\sigma_D,\sigma_I,I\left(\mathbf{x}\right)\right) \quad = \quad g\left(\mathbf{x},\sigma_I\right) * \begin{bmatrix} I_x^2\left(\mathbf{x},\sigma_D\right) & I_xI_y\left(\mathbf{x},\sigma_D\right) \\ I_xI_y\left(\mathbf{x},\sigma_D\right) & I_y^2\left(\mathbf{x},\sigma_D\right) \end{bmatrix}. \tag{4.7}$$

Here σ_D is referred to as the differentiation scale and σ_I is the integration scale. Convolution with the Gaussian operator with scale σ_D performs the scale space operation. Convolution with the Gaussian operator with scale σ_I effectively performs windowed integration of the elements of the matrix. The second moment matrix is a positive definite, symmetric matrix and can be seen as a local estimate of the covariance matrix of an image.



A test pattern, $I(\mathbf{x})$.

$I(\mathbf{x}, \sigma)$.

$I_{x^2}(\mathbf{x}, \sigma)$.

$I_{xy}(\mathbf{x}, \sigma)$.

$I_{y^2}(\mathbf{x}, \sigma)$.

$\left|\frac{\partial^2 I(\mathbf{x},\sigma)}{\partial \mathbf{x}^2}\right|$.

Figure 4.6: Computing the Determinant of Hessian. The first row shows a test pattern and the same pattern filtered with a Gaussian. The second row and left image of the third row show the second order partial derivatives of the filtered image (the elements of the Hessian matrix, Equation 4.6). The last image shows the Determinant of the Hessian matrix.

The Harris operator is computed from the second moment matrix as follows,

$$H\left(\mathbf{x},\sigma_D,\sigma_I,I\left(\mathbf{x}\right)\right)=\left|\mu\left(\mathbf{x},\sigma_D,\sigma_I,I\left(\mathbf{x}\right)\right)\right|-k\left(\mathrm{trace}\left(\mu\left(\mathbf{x},\sigma_D,\sigma_I,I\left(\mathbf{x}\right)\right)\right)\right)^2, \quad (4.8)$$

where k is often set to 0.04. The measure computed by this operator is commonly referred to as the Harris cornerness measure. The cornerness measure is large when the eigenvalues of the second moment matrix are both large, indicating significant change in the image in two orhtogonal directions. Computing equation 4.8 is less computationally demanding than computing the eigenvalues directly.

Figure 4.7 illustrates the process of applying the Harris operator. Note that this operator requires additional filtering steps compared to the determinant of Hessian operator. It can be seen that edge responses are suppressed very effectively and that the corner locations are distinguished from the surrounding image with high contrast.

The Harris corner detector was described in [21]. It forms the basis of some popular modern affine covariant feature extractors [4, 43, 45].

4.2.2 Covariance Properties

Salient features are selected by finding the extrema of saliency maps. This process only generates point coordinates for each feature and does not specify a support region directly. The position of extrema produced by saliency operators are translation and rotation (isometrically) covariant. Translation covariance arises naturally from the fact that the convolution operation is translation invariant. The saliency operators listed above are rotationally symmetric and therefore rotation covariant (rotationally invariant at the centre of rotation). Saliency operators are very sensitive to a change in scale. They are sensitive to a change in view angle to a limited degree due to the relative change in scale of image structures.

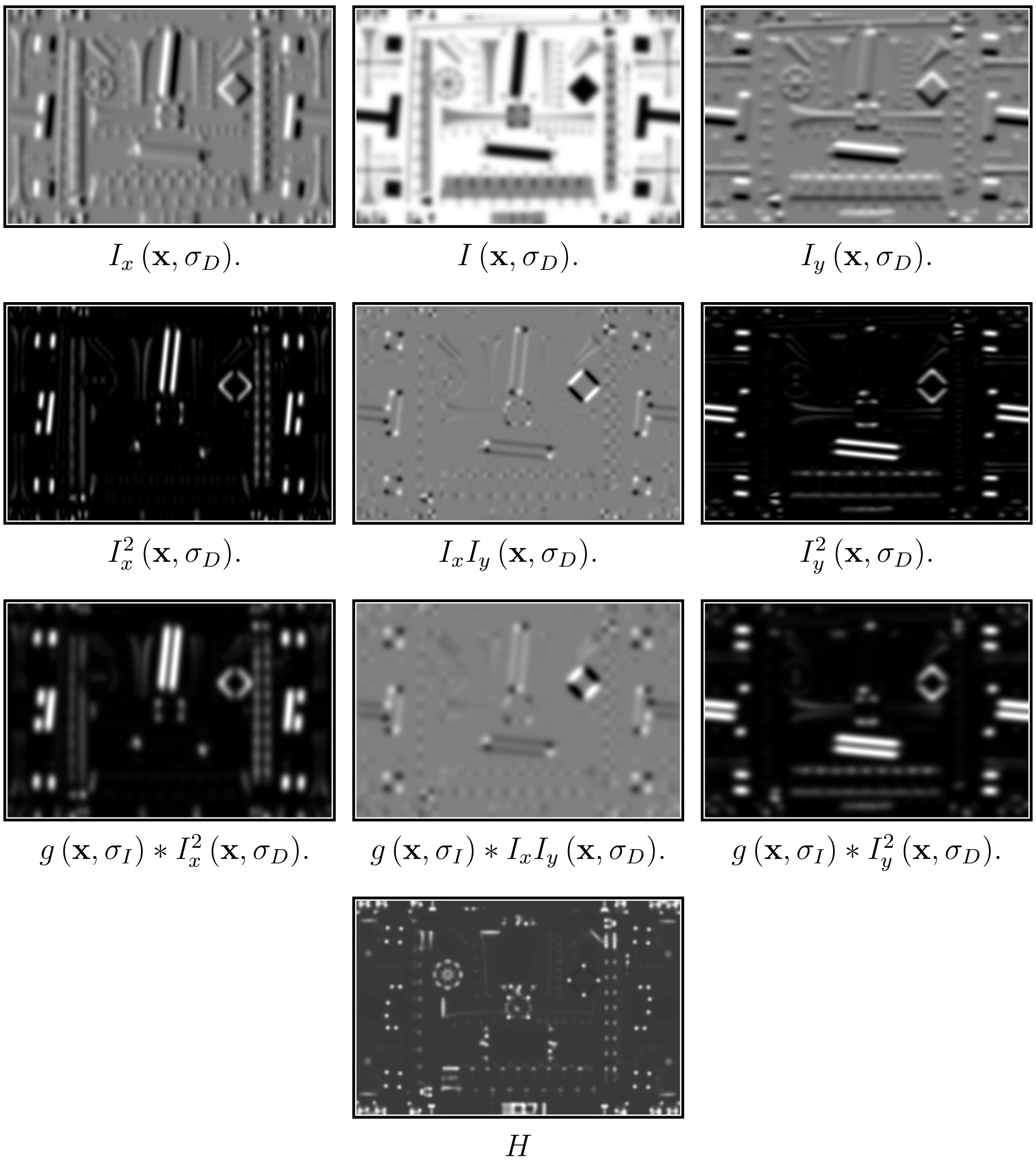

$I_x(\mathbf{x}, \sigma_D)$.
$I(\mathbf{x}, \sigma_D)$.
$I_y(\mathbf{x}, \sigma_D)$.
$I_x^2(\mathbf{x}, \sigma_D)$.
$I_xI_y(\mathbf{x}, \sigma_D)$.
$I_y^2(\mathbf{x}, \sigma_D)$.
$g(\mathbf{x}, \sigma_I) * I_x^2(\mathbf{x}, \sigma_D)$.
$g(\mathbf{x}, \sigma_I) * I_xI_y(\mathbf{x}, \sigma_D)$.
$g(\mathbf{x}, \sigma_I) * I_y^2(\mathbf{x}, \sigma_D)$.
H

Figure 4.7: The process of computing the Harris cornerness measure. The first row shows the differentiation scale image in the centre, with its first order partial derivative images on either side. The second row shows products of the derivative images. The third row shows the images in the second row filtered with an integration scale Gaussian. The last image shows the Harris cornerness measure computed from the images in the third row using Equation 4.8.

4.3 Affine Feature Extraction

4.3.1 Scale Selection

Early scale conscious feature extraction methods simply extract maxima from the saliency map for every discrete scale level in a scale space pyramid. The result is a very large set of features with large groups of similar features. Most of these features are not likely to yield appropriate point correspondences when matching against a similar set from a different viewpoint. An example can be found in [16], where this method is used to match a low resolution image to a high resolution image. Extracting multi-scale features in this way is not sufficient to produce scale invariant features, but is often used by other methods as a first step in exploring the scale space.

A more successful method of producing scale invariant features is to select characteristic scale features. In [34] it is proposed that some combination of scale-normalised derivatives (a scale response function) computed in the vicinity of an image structure will assume a local maximum at the scale corresponding to the structure size. The scale response function is thereby used as a correlation detector to select a characteristic scale. The characteristic scale selected is therefore directly related to the characteristic size of the image structure. The simplest method of characteristic scale selection is to locate scale space maxima – points that represent local maxima of the saliency map in the spatial and scale dimensions. This is consistent with the proposition in [34], since saliency functions are usually composed from normalised derivatives. Example implementations have been published describing scale space non-maximum suppression using the Laplacian [34], Difference of Gaussians [38] and determinant of Hessian [5] functions.

The scale space maxima approach was extended in [43] by locating points that are local maxima of the Harris operator in the spatial dimensions and maxima of the scale-normalised Laplacian in the scale dimension. Utilising different functions for selecting scale and for locating spatial position makes it possible to use the functions that are most stable in each domain. For example, the Harris operator produces stable maxima in the spatial domain, but rarely produces stable maxima in scale space. The Laplacian is useful for scale selection, but is not as effective for feature selection, since it often results in the selection of ambiguous points along nearly straight edges.

Selecting scale space maxima using simple 3D non-maximum suppression in scale space pyramids has a few drawbacks. This method often rejects points that are useful features as a result of the limited sampling density of the scale space pyramid. Non-maximum suppression algorithms require that successive pyramid levels be sampled consistently in the spatial dimensions. This places restrictions on how the scale space can be sampled and limits the savings that can be made by progressive down-sampling (see Section 4.5.2).

An iterative scale selection method was proposed in [43] which does not suffer from the above issues. It offers higher accuracy scale selection at the cost of greater computation time. The algorithm is initialised by selecting multi-scale points using the Harris or Determinant of Hessian operator and each point is iteratively adapted. At each iteration, i, the following two steps are executed:

1. The new scale σ_i at which the Laplacian of Gaussian achieves a local maximum over scales is sought in the range $[0.7\sigma_{i-1}, 1.4\sigma_{i-1}]$ at the position $\mathbf{x}_{i-1}$. If no local maximum is found the feature is rejected.

2. The new spatial point $\mathbf{x}_i$ nearest $\mathbf{x}_{i-1}$ is found at which the Harris operator evaluated at scale σ_i achieves a maximum.

This continues until the point converges in scale space.

4.3.2 Shape Adaptation

The scale adaptation processes presented in the preceding chapter operate by computing the saliency map for the three parameter scale space, $I(\mathbf{x}, \sigma)$. It is desirable to extend the feature extraction problem to include the shape parameters q and θ. Simply extending the image space to an affine scale space, $I(\mathbf{x}, \sigma, q, \theta)$, and evaluating the saliency map over the whole space would require an excessive amount of processing effort (isotropic Gaussian scale space is already quite expensive to compute). Fortunately this is not required, since the shape parameters do not have a drastic effect on the location of extrema of saliency operators. It is sufficient to only make use of the local space around characteristic scale features to compute the feature shape. It is important to note that the local shape estimation process does not estimate the shape of the imaged surface (this is not possible), but instead computes a shape estimate based on the local intensity distribution.

The most effective modern shape estimation methods derive from [35], which uses the second moment matrix to iteratively measure local shape. In [35], the Gaussian scale space is extended to affine Gaussian scale space. The affine Gaussian operator is of the form,

$$g(\mathbf{x}, \Sigma) = \frac{1}{2\pi|\mathbf{\Sigma}|} e^{\frac{-\mathbf{x}^\top \mathbf{\Sigma}^{-1}\mathbf{x}}{2}},$$

where $\mathbf{\Sigma}$ is a positive definite symmetric matrix known as the covariance matrix and $|\mathbf{\Sigma}|$ is the determinant of $\mathbf{\Sigma}$. The affine scale space is,

$$I(\mathbf{x}, \mathbf{\Sigma}) = g(\mathbf{x}, \Sigma) * I(\mathbf{x}).$$

The affine second moment matrix computed in affine scale space is defined as,

$$\begin{aligned}\mu(\mathbf{x}, \mathbf{\Sigma}_D, \mathbf{\Sigma}_I) &= |\mathbf{\Sigma}_D|\, g(\mathbf{x}, \mathbf{\Sigma}_I) * \mathbf{D}_A, \\ \mathbf{D}_A &= \begin{bmatrix} I_x^2(\mathbf{x}, \mathbf{\Sigma}_D) & I_x I_y(\mathbf{x}, \mathbf{\Sigma}_D) \\ I_x I_y(\mathbf{x}, \mathbf{\Sigma}_D) & I_y^2(\mathbf{x}, \mathbf{\Sigma}_D) \end{bmatrix},\end{aligned}$$

with $\mathbf{\Sigma}_D$ and $\mathbf{\Sigma}_I$ differing only in scale.

Affine adaptation is performed by iteratively computing the second moment matrix as,

$$\mathbf{M}_i = \mu(\mathbf{x}, k_D\mathbf{M}_{i-1}, k_I\mathbf{M}_{i-1}),$$

where i is the iteration number, k_D is chosen to maximise the value of the Laplacian at $\mathbf{x}$, k_I is chosen so that the minimum eigenvalue of $k_I\mathbf{M}$ remains constant during iterations and $\mathbf{M}_0 = \mathbf{I}$. It is shown in [35] that if $\mathbf{M}$ is computed as above, then it converges such that, for sufficiently large n,

$$\mu(\mathbf{x}, k_D\mathbf{M}_n, k_I\mathbf{M}_n) \approx \mathbf{M}_n.$$

The resulting matrix $\mathbf{M}$ is covariant under affine transformations of the image. This method adapts the scale and shape components while the feature position is kept fixed at its initial position.

The method presented in [4] applies a normalising affine transformation to a local image region, instead of adapting the parameters of affine scale space. Integration scale and differentiation scale of the second moment matrix operator are set proportional to the scale at which the feature is detected. At each iteration of the algorithm the local image region around the selected feature is transformed using the inverse square root of the second moment matrix computed during the previous iteration (initially $\mathbf{I}$). The second moment matrix is then computed again from the normalised image region, using radially symmetric Gaussian kernels, and is normalised to have a determinant of 1. This continues until the measured normalised second moment matrix is sufficiently close to the identity matrix. The final shape transformation is the composition of all the normalisation transformations applied during adaptation. The method published in [4] is more easily implemented and more efficient than the method of [35], but only adapts the shape component while leaving the scale and position fixed. This method is also applied to both Harris and Determinant of Hessian features in [45].

A more complete algorithm is presented in [43] that updates the integration scale, differentiation scale and feature location at each iteration, before computing the second moment matrix. It is the affine extension to the scale covariant algorithm presented in the same article. A summary of the algorithm is presented in Algorithm 4.1. A measure of local shape isotropy is defined based on the eigenvalues $\lambda_{\min}$, $\lambda_{\max}$ of the second moment matrix as,

$$Q(\mu) = \frac{\lambda_{\min}(\mu)}{\lambda_{\max}(\mu)}. \tag{4.9}$$

Adaptation concludes when Q is sufficiently close to 1. This is a more computationally expensive algorithm than the method of [4]. It is the only algorithm that allows for a change in scale and position as the shape is adapted. In the evaluation presented in [45], the authors chose to use a method most similar to [4], and the method in [43] was not evaluated. The performance of this algorithm is therefore unknown, however the author's preference for the method in [4] indicates that perhaps the method in [43] does not produce superior results.

4.3.3 Orientation Selection

The extractors found in the literature do not assign an orientation to the features, and instead defer this task to the descriptors. This appears to be a reasonable choice, since some descriptors are rotation invariant and others are not. The extractor evaluations (See Chapter 3) are also not affected by feature orientation. The SIFT descriptor [39] and descriptors derived from it make use of a separate

Algorithm 4.1: Summary of the Affine Adaptation Algorithm in [43].

```
4.1.1  begin
4.1.2      $\mathbf{A}_0 \leftarrow \mathbf{I}$.
4.1.3      repeat
4.1.4          $I_n(\mathbf{x}) \leftarrow I_i(k\mathbf{A}_i\mathbf{x} + \mathbf{x}_c)$.
4.1.5          Select integration scale $\sigma_{Ii}$ that maximises LoG over scale at
               point $\mathbf{x}_{i-1}$.
4.1.6          Select differentiation scale that maximises $Q$ (Eqn. 4.9) from
               the range $\sigma_{Di} \in [0.5, 0.75]\,\sigma_{Ii}$.
4.1.7          Find new spatial point $\mathbf{x}_i$ nearest $\mathbf{x}_{i-1}$ at which the Harris
               operator achieves a maximum.
4.1.8          $\mu_i \leftarrow \mu(\mathbf{x}_i, \sigma_{Di}, \sigma_{Ii})$.
4.1.9          $q \leftarrow Q(\mu_i)$.
4.1.10         $\mathbf{A}_i \leftarrow \mu_i^{-\frac{1}{2}}\mathbf{A}_{i-1}$.
4.1.11         $\mathbf{A}_i \leftarrow \lambda_{\max}^{-1}(\mathbf{A}_i)\,\mathbf{A}_i$.
4.1.12     until $q > Q_T$
4.1.13 end
```

orientation selection process, which is discussed below.

The orientation selection method presented in [39] operates as follows. First the feature transformation is computed using the saliency map, scale selection and affine adaptation as described in the preceding sections of this chapter. A normalised image is then computed using the feature transformation. The orientation θ and magnitude m of the image gradients are computed at every pixel in the normalised image, resulting in corresponding images $\Theta(\mathbf{x})$ and $M(\mathbf{x})$. A Gaussian function with scale parameter set to 1.5 times the scale of the normalised feature is used to weight the values of the gradient magnitude image, $M(\mathbf{x})$. This progressively reduces the contribution of gradients further from the feature centre. The values of $M(\mathbf{x})$ are then accumulated in a histogram according to the corresponding angle in $\Theta(\mathbf{x})$. The histogram has 36 bins for the 360° range of angle values. The maximum value in this histogram corresponds to the dominant gradient orientation of the normalised image. The angle corresponding to the maximum value in the histogram, as well as all local maxima within

80% of the maximum, are selected as characterisitic orientations for the feature. Multiple orientations may therefore be assigned to each feature. The precision of each selected angle is refined by fitting a parabola to the three values in close proximity to the maximum and computing the location of the vertex.

4.4 A Framework for Affine Salient Feature Extraction

In the literature, feature extractors are usually discussed in terms of a specific solution, including saliency operator, scale selection method and affine adaptation method. A number of the saliency map-based methods make use of a common overarching procedure, varying only in the functions used to estimate each component of the feature affine transformation. A general adaptation framework, independent of specific estimation functions, is required in order to compare different estimation functions. In this section, the design of a generalised salient feature extractor and affine adaptation algorithm is presented. This algorithm is employed in the experimental evaluation of the various new techniques developed in later chapters.

The existing affine covariant feature extraction algorithms essentially follow a similar overarching design. They consist of the following modules:

1. Feature location by finding local maxima in a saliency map.
2. Feature scale selection by finding a maximum over scales in a function of normalised derivatives.
3. Feature affine shape estimation using a second order shape measure.

4. Orientation selection.

These modules are combined in different ways by different authors. Most algorithms consist of an initialisation stage, where initial feature points are found in scale space, and an adaptation stage, where each feature is adapted to a point and shape that is affine covariant (for example [4, 43, 45]). The modular view of adaptation algorithms presented above allows the design of an adaptation algorithm independent of the various measures used.

The generalised affine covariant feature extractor algorithm is listed in Algorithm 4.2. Section 4.4.1 discusses the various subroutines employed by the algorithm and Section 4.4.2 discusses the algorithm as a whole.

4.4.1 Subroutines and Constants

Generic Functions

One of the design objectives is to define an algorithm that can use any set of parameter estimation functions. The generic estimator functions listed below are used in the algorithm definition to achieve this objective. A specific implementation would substitute specific functions for each of these generic forms.

$I_e\left(\mathbf{x},\sigma_i\right) \leftarrow \mathrm{E}\left(I\left(\mathbf{x}\right),\sigma_i\right)$ represents the saliency operator. It accepts an input image $I\left(\mathbf{x}\right)$ and scale σ_i, and produces the saliency map $I_e\left(\mathbf{x},\sigma_i\right)$. Alternatively, σ_i may be a set of scales and $I_e\left(\mathbf{x},\sigma_i\right)$ a set of images, one for each element in σ_i. This produces a scale space pyramid of saliency maps.

$r_\sigma \leftarrow \mathrm{S}\left(I\left(\mathbf{x}\right),\mathbf{x}_i,\sigma_i\right)$ is the scale response operator. The inputs are an image and the scale space coordinates at which to compute the response r_σ.

Algorithm 4.2: Generalised Feature Extraction Algorithm.

```
4.2.1  Function $f_A \leftarrow \mathrm{EXTRACT}\left(I_i\right)$
       Input:
       $I_i\left(\mathbf{x}\right)$ – An image.
       Output:
       $f_A = \{\mathbf{H}_0, \mathbf{H}_1, \ldots, \mathbf{H}_n\}$ – A vector of features in the form of a
       normalisation transformation matrix for each feature.
4.2.2  begin
4.2.3      $\sigma_p \leftarrow \{\sigma_0, \sigma_1, \ldots, \sigma_{n_s}\}$.
4.2.4      $I_e\left(\mathbf{x}, \sigma_p\right) \leftarrow E\left(I_i\left(\mathbf{x}\right), \sigma_p\right)$.
4.2.5      $f_m \leftarrow \mathrm{MSMAXIMA}\left(I_e\left(\mathbf{x}, \sigma_p\right), T\right)$.
4.2.6      $f_s \leftarrow \mathrm{SCALESELECT}\left(\mathrm{S}, f_m\right)$.
4.2.7      $f_s \leftarrow \mathrm{TOPN}\left(f_s, n_f\right)$.
4.2.8      $f_A \leftarrow \emptyset$.
4.2.9      foreach $f_j \in f_s$ do
4.2.10         $\left(\mathbf{x}_j, \sigma_j\right) = f_j$.
4.2.11         $i \leftarrow 0$.
4.2.12         $\mathbf{A}_j \leftarrow \mathbf{I}$.
4.2.13         $q \leftarrow Q_l + \epsilon$.
4.2.14         while $\left(i < n_i\right) \cdot \left(q < Q_h\right) \cdot \left(q > Q_l\right)$ do
4.2.15             $k \leftarrow \lambda_{\max}^{-1}\left(\mathbf{A}_i\right)$.
4.2.16             $I_n\left(\mathbf{x}\right) \leftarrow I_i\left(k\mathbf{A}_j\mathbf{x} + \mathbf{x}_j\right)$.
4.2.17             $\mathbf{A}_u \leftarrow \mathrm{A}\left(I_n\left(\mathbf{x}\right), \mathbf{0}, k\sigma_j\right)$.
4.2.18             $\mathbf{A}_j \leftarrow \mathbf{A}_u^{\frac{1}{2}}\mathbf{A}_j\mathbf{A}_u^{\frac{1}{2}}$.
4.2.19             $q \leftarrow \mathrm{Q}\left(\mathbf{A}_u\right)$.
4.2.20         end
4.2.21         if $q > Q_r$ then
4.2.22             $k \leftarrow \lambda_{\max}^{-1}\left(\mathbf{A}_i\right)$.
4.2.23             $I_n\left(\mathbf{x}\right) \leftarrow I_i\left(k\mathbf{A}_j\mathbf{x} + \mathbf{x}_j\right)$.
4.2.24             $\mathbf{R}_j \leftarrow \mathrm{R}\left(I_n\left(\mathbf{x}\right), \mathbf{0}, k\sigma_j\right)$.
4.2.25             $\mathbf{H}_j \leftarrow \begin{bmatrix} \sigma_j\mathbf{R}_j\mathbf{A}_j & -\mathbf{x}_j \\ \mathbf{0} & 1 \end{bmatrix}$.
4.2.26             $f_A \leftarrow \{f_A, \mathbf{H}_j\}$.
4.2.27         end
4.2.28     end
4.2.29 end
```

$\mathbf{A} \leftarrow \mathrm{A}\left(I\left(\mathbf{x}\right), \mathbf{x}_i, \sigma_i\right)$ represents the shape estimator function. The inputs are an image and the scale space coordinates at which to compute the shape. The output is a positive definite symmetric 2×2 shape matrix.

$\mathbf{R} \leftarrow \mathrm{R}\left(I\left(\mathbf{x}\right), \mathbf{x}_i, \sigma_i\right)$ is the orientation selection function. The inputs are an image and the scale space coordinates around which to compute a characteristic orientation. It returns a rotation matrix, or set of rotation matrices $\mathbf{R}$.

$f_c \leftarrow \mathrm{SCALESELECT}\left(\mathrm{S}, f_m\right)$ is the generic scale selection algorithm. The input, S, is a scale response function and f_m is a set of scale space feature coordinates. The output, f_c, is a smaller set of refined characteristic scale features in the same format as f_m.

Subroutines

The algorithm makes use of several trivial subroutines. These are listed below.

$q \leftarrow \mathrm{Q}\left(\mathbf{M}\right)$ returns the isotropy measure of the 2×2 symmetric matrix $\mathbf{M}$, as defined in Equation 4.9 [43].

$e \leftarrow \lambda_{\max}\left(\mathbf{M}\right)$ returns the largest eigenvalue of square matrix $\mathbf{M}$.

$f \leftarrow \mathrm{MSMAXIMA}\left(I_e\left(\mathbf{x}, \sigma_p\right), T\right)$ extracts the local 2D maxima in each level, $\sigma_i \in \sigma_p$, of map pyramid $I_e\left(\mathbf{x}, \sigma_p\right)$, with discrete scale levels $\sigma_p = \{\sigma_0, \sigma_1, \ldots, \sigma_n\}$. Points with a value lower than threshold T are not selected. A set of scale space coordinates, $f = \{\left(\mathbf{x}_0, \sigma_0\right), \left(\mathbf{x}_1, \sigma_0\right), \ldots, \left(\mathbf{x}_m, \sigma_n\right)\}$, are returned.

$f_s \leftarrow \mathrm{TOPN}\left(f_i, n\right)$ selects the n features from set f_i that have the highest scale-normalised saliency value. If f_i contains fewer than n features, then $f_s = f_i$.

Constants

The following constants are used in the algorithm. Where values are given, they were chosen empirically to achieve a balance between computation time and feature quality.

T. The threshold used during maxima extraction to reject weak points. This should be chosen depending on the particular saliency function in use.

n_f. The maximum number of initial characteristic scale features to adapt. This should be chosen according to needs of the application.

Q_l. The minimum allowed isotropy measure. Features producing a shape matrix estimate with isotropy lower than this are discarded. This threshold is used to avoid adapting features that are on straight edges and are not corners or blobs. Implementations used in this thesis use $Q_l = 0.05$.

Q_h. The isotropy measure convergence threshold. Features producing a shape matrix estimate with isotropy higher than this are considered to have converged. This thesis uses 0.98.

Q_r. Features with an isotropy measure below this threshold, after the maximum number of iterations have been reached, are rejected. Implementations used in this thesis use $Q_r = 0.5$.

n_i. The maximum number of iterations allowed for the affine shape estimation process. It was determined, through experimentation, that allowing more than 8 iterations does not improve extractor performance significantly, and only increases computation time. The value of n_i is therefore set to 8 in this thesis.

4.4.2 Discussion

The algorithm extracts an initial set of multi-scale features by computing the multi-scale saliency map using operator E and extracting 2D maxima from this map (lines 4.2.3 - 4.2.5). Scale selection is performed using the scale selection routine SCALESELECT and scale response operator S in line 4.2.6. In line 4.2.7 the number of features is limited by removing the features with lower saliency response and keeping only n_f features (by means of the TOPN function). The purpose of this step is to limit the processing time and memory requirements during the adaptation step. It makes it possible to control the number of features more specifically than simply applying a global, predetermined threshold to the saliency map.

The rest of the extraction algorithm adapts each feature individually. The affine shape matrix is initialised with the identity matrix, and the isotropy measure is initialised with a value between the lower and upper isotropy thresholds (lines 4.2.12, 4.2.13). Each iteration involves normalising the local image region around the feature (lines 4.2.15, 4.2.16)), measuring the feature's affine shape using the shape estimator A (line 4.2.17), updating the normalisation matrix (line 4.2.18), and checking the shape isotropy (line 4.2.19, 4.2.14) to determine whether convergence has been reached.

Image normalisation is performed by transforming the original image by a modified version of the latest normalisation matrix and translating the feature point to the origin. The shape matrix is modified such that its largest eigenvalue is 1 in order to prevent aliasing. The scale factor used to modify the shape matrix is incorporated in the shape estimation process accordingly. The formula used for updating the normalisation matrix (line 4.2.18) ensures that the matrix remains symmetric and that no extra rotation component is introduced. At the end of each iteration the isotropy of the measured shape is checked. If the isotropy measure is too small, then the feature is too elongated and is probably a nearly straight line. If the shape is almost perfectly isotropic, then adaptation has converged and further adaptation will have little effect. Most features reach convergence after only a few iterations, but not all features are guaranteed to converge quickly. The number of iterations is limited to prevent getting stuck on features that do not converge stably and quickly.

If affine shape adaptation completes successfully (i.e., the last measured isotropy is above Q_r), then the feature orientation is selected and the final feature transformation composed. First the feature support region is normalised using the previously computed affine shape (lines 4.2.22 and 4.2.23) similar to the normalisation step in the affine shape estimation process. The orientation is then selected using R (line 4.2.24). Finally, the complete feature transformation matrix is constructed from all the measured parameters in line 4.2.25. If more than one rotation matrix was returned by R, then one feature transformation matrix is constructed for each rotation matrix.

4.5 Implementing Image Operators

4.5.1 The Gaussian Filter and its Derivatives

The Gaussian function (equation 4.1) is non-zero for all real coordinates. When used as a digital convolution filter, the convolution kernel must be computed from the Gaussian function over a limited support region. Truncating to a coordinate range of $(-3\sigma, 3\sigma)$ results in a filter with over 99% of the energy of the function with infinite support. The derivatives of the Gaussian may require a larger support region, depending on the sensitivity requirements of the application. Convolution with a 2D Gaussian kernel has complexity $O(n\sigma^2)$, where n is the number of image pixels. Computing a scale space representation of an image using 2D Gaussian convolution is computationally expensive. Fortunately, the 2D Gaussian function and its derivatives have several properties that allow more efficient implementation.

Separability

The 2D Gaussian function may be expressed as the convolution of two separate 1D Gaussian functions,

$$g(x, y, \sigma) = g(x, \sigma) * g(y, \sigma). \tag{4.10}$$

In other words, the Gaussian image filter operation may be implemented as a two stage process where a 1D filter is applied along each principal direction. It is said that the Gaussian kernel is separable. This reduces the filter complexity to $O(n\sigma)$.

Recursive Implementation

The 1D Gaussian filter and its derivatives may be implemented in the form of a recursive filter with a number of filter taps that is fixed and independent of σ. Two slightly different methods for constructing a recursive implementation of the Gaussian filter have been published in [14] and [59]. The complexity of the recursive separable implementation of the Gaussian filter is linear in the number of image pixels and independent of filter scale. For very small scales, separable convolution filters require less time to compute than the equivalent recursive filter. The time required by convolution filters quickly exceeds that of the recursive implementation as scale increases. The exact scale at which the recursive version becomes more efficient depends on the number of filter taps used in the recursive implementation.

Repeated filtering

Filtering an image with two Gaussian filters with scales σ_1 and σ_2 is equivalent to filtering with a single Gaussian with scale $\sigma_{1+2} = \sqrt{\sigma_1^2 + \sigma_2^2}$. When using recursive filters, this fact is of little importance. When using convolution filters to construct a scale space pyramid, for example, the total processing time may be reduced by producing each scale level from the preceding scale level, instead of filtering the original image. This allows the use of smaller scale filters.

Derivatives

As explained in Section 4.1.2, the derivatives of a scale space image may be computed by taking the derivatives of the image after the convolution with the Gaussian, or by convolving with the derivatives of the Gaussian (see Equation 4.3). The implication is that a scale space derivative image may be produced either by filtering with a Gaussian derivative filter, or by filtering with a Gaussian and subsequently computing the derivative. In the case where multiple different derivatives are to be computed at the same scale (as is common in computing saliency functions), it is more efficient to filter with a Gaussian and to compute the derivatives separately, since Gaussian derivative filters are more computationally expensive than the difference filters used to compute image derivatives. On the other hand, using Gaussian derivative filters produces slightly superior results, since the derivative is computed analytically, rather than by means of a digital difference filter.

4.5.2 Scale Space

The Gaussian scale space of an image is most commonly implemented by computing a scale space pyramid – a set of images that each sample the scale space at one scale. It is most practical to sample the scale space at $\sigma = 2^k$, where k is a regular series of positive values. At lower scales, a small increase in scale results in a significant reduction in detail. At higher scales, in contrast, little detail is present so that a small change in scale results in little change in detail. Sampling using an exponential series of sampling scales results in a set of samples where the level of detail decreases more monotonically.

It is common to down-sample the pyramid in the spatial dimensions as scale increases, in order to reduce the computation time required to produce the pyramid. The down-sampling may be performed in several ways. The simplest is to down-sample every level, resulting in a monotonic reduction in sampling density. An alternative is to down-sample after each scale octave (at scales $\sigma = 2^i$ where $i \in \mathbb{N}$) and to add an extra level before and after each octave that are sampled at the same spatial resolution as the octave. This method is used where there is a need for consistent spatial sampling over multiple levels. An example is illustrated in [39], where 3D non-maximum suppression is applied.

4.5.3 Saliency Operators

The saliency operators introduced in Section 4.2.1 are composed of linear combinations of scale space derivatives. The operator output may be computed by computing the derivative images as described in Section 4.5.1 and then computing the weighted sum or product of the corresponding pixels in the derivative images. Examples are illustrated in Figures 4.6 and 4.7 where the Determinant of Hessian and Harris cornerness operators are computed.

4.6 Chapter Summary

This chapter reviews saliency map-based feature extraction, scale selection, shape estimation and orientation selection. A saliency map provides a method for analysing the local information content in images. Regions of distinct curvature produce extrema in saliency maps.

Saliency operators are inherently scale sensitive. Gaussian scale space is introduced as a platform for multi-scale analysis. Saliency operators are then reviewed in the context of Gaussian scale space. Among the many possible saliency operators, the Harris and Determinant of Hessian operators provide the most stable features. Of these two, the Determinant of Hessian is the simplest to compute and has been used to produce the most repeatable feature extractor.

Extracting features using a saliency map provides a set of scale space coordinates for each feature, but does not provide a scale, shape or orientation directly. Methods for characteristic scale selection attempt to find for each feature a maximum over scale of various functions of normalised scale space derivatives. Such a maximum indicates a characteristic scale. An affine covariant shape may be computed for a local image region by iteratively measuring the second moment matrix. Several similar shape estimation procedures are reviewed. The orientation of a feature is computed last, after the local image region around the feature has been normalised in terms of scale and affine shape.

A generalised framework for extracting affine covariant features by means of a saliency map and affine adaptation is presented. This framework accommodates the use of different saliency operators, scale selection methods, affine shape estimation methods, and orientation selection methods. It will be used in later chapters to compare the performance of different alternative components of the affine adaptation process.

The last section of this chapter briefly discusses the implementation of the Gaussian operator, scale space and scale space derivative operations.

Chapter 5

The Salient Feature Scale Space Primal Sketch

The concept of scale plays an important role in wide baseline vision. Chapter 4, Section 4.3.1 discusses methods proposed in the literature to select a characteristic scale for each local image feature. Characteristic scale selection is an effective method used to deal with the scale problem in the domain of robust local feature extraction.

The scale space primal sketch has in the past been a useful tool for multi-scale analysis of features and for scale selection [32], but is not used in modern feature scale selection methods. This chapter develops a method for computing a discrete representation of the scale space primal sketch of local image features. This discrete primal sketch is used to implement a more efficient and effective method for scale selection by combining the sketch with modern feature extraction and scale selection techniques.

Section 5.1 introduces the scale space primal sketch. The concept is extended to the sketch of saliency map features in scale space and the motivation for exploring this technique is examined. An algorithm for building a feature scale space primal sketch is developed in Section 5.2. In Section 5.3, the primal sketch is used to create a scale selection algorithm. The performance of this algorithm is compared to existing methods. Section 5.4 concludes the chapter with a summary.

5.1 The Scale Space Primal Sketch

The scale space primal sketch was introduced in [31, 32]. It is the graph consisting of the loci of functional primitives over scale. In [31, 32] the primitives used are grey level blobs, or local extrema in the image intensity function. The scale space primal sketch was used to select the characteristic scale of grey level blobs by finding the scale at which the blob assumes the maximum volume. Other applications include an edge extraction method, where the scale of the edge finding operator is selected automatically using the primal sketch.

The scale space primal sketch concept may be extended to extrema of the derivatives of scale space and to saliency maps. In terms of the primal sketch representation, the only modification is that the primitives are chosen to be extrema of saliency operators. This representation will be referred to as the scale space feature sketch. The method for computing the scale space feature sketch representation must be derived from the behaviour of features in scale space and the methods for computing the scale space itself.

The motivation for exploring the scale space feature sketch as a multi-scale feature analysis tool is discussed below.

5.1.1 Motivation

The existing feature characteristic scale selection methods (reviewed in Chapter 4, Section 4.3.1) use either scale space maxima or an iterative search strategy. Each of these methods has limitations. Using a scale space feature sketch may provide a means for avoiding these limitations. The feature sketch also provides a multi-scale representation of features; however, the discussion in this chapter will focus on the application of characteristic scale selection.

Pyramid Construction

The scale space maxima method imposes restrictions on how the scale space pyramid can be constructed. The non-maximum suppression technique commonly used for finding maxima in digitally sampled data requires the data to be sampled uniformly. The scale space pyramid must therefore be constructed such that adjacent layers are sampled at the same spatial resolution. Either the whole pyramid must be sampled at the same spatial resolution, or extra layers must be generated each time the pyramid is down-sampled. These restrictions limit the savings, in terms of processing time, that can be made by progressively down-sampling the pyramid.

The scale space feature sketch is a graph structure in which each node is a feature, denoted by its coordinates. The structure of this graph does not depend on the method used to extract the feature coordinates, and therefore does not impose any restrictions on how they are computed. The individual features need only be extrema in the spatial directions, and therefore extracting a feature on one pyramid layer does not require processing any other layer. Section 5.2 describes an algorithm for constructing the scale space feature sketch from a set of multi-scale features without any additional image processing. In short, none of the feature sketch graph, features or graph construction algorithms depend on the structure of the scale space pyramid.

Accuracy

The accuracy of the scale space maxima method is limited by the scale resolution of the pyramid. If the scale space were sampled infinitely densely, then finding scale space maxima would be equivalent to tracking the evolution of spatial extrema over scale and selecting points where the spatial maxima also reached a maximum over scale. In a discrete pyramid, however, the spatial maxima may be displaced by a distance of several pixels between consecutive scale layers. This can result both in spurious maxima being detected and in true maxima being missed by the non-maximum suppression process. The scale space feature sketch, by definition, is a discrete representation of the evolution of spatial extrema over scale. Therefore, if the feature sketch is computed correctly, it addresses the above problem directly by keeping track of spatial extrema over scale.

Different Functions for Spatial and Scale Selection

The literature indicates that selecting points where the saliency function assumes a local maximum over the spatial dimensions, and where the Laplacian assumes a local maximum over scales, results in good quality characteristic scale features [43, 45]. This is because the Laplacian acts as a matched filter for structure size.

This scale selection scheme may be implemented using the scale space feature sketch. All nodes in the feature sketch are at extrema of the saliency map over the spatial dimensions. By evaluating the Laplacian at the scale space position of each feature in the sketch, it is possible to select a feature where the Laplacian assumes a local maximum in the graph. In this way the Laplacian is strictly evaluated along the feature locus in scale space and provides a structure size matching strength for each feature. It is hypothesised that this method will yield superior results. The scale space feature sketch-based scale selection method is presented in detail and evaluated in Section 5.3.

5.2 The Scale Space Feature Sketch Graphing Algorithm

5.2.1 Objective

The objective is to develop an algorithm to efficiently extract the scale space sketch of the spatial extrema of a saliency function. This objective is formalised as follows: Given the Gaussian scale space image, $I(\mathbf{x}, \sigma)$, a scale space saliency map, $D(\mathbf{x}, \sigma)$ is computed by applying some saliency operator to the image,

$$D(\mathbf{x}, \sigma) = D(I(\mathbf{x}, \sigma)).$$

Select a point, $\mathbf{x}_m$, at the lowest available scale, σ_0, in $D(\mathbf{x}, \sigma)$ where the value of D assumes a locally extreme value over the spatial dimensions (it does not need to be a maximum in the scale direction). As the scale dimension is traversed smoothly, the spatial location of the extreme point will vary smoothly, according to the properties listed in Chapter 4, Section 4.1.3. Define the locus of an extremal point in scale-space as $\mathbf{x}_\sigma = s(\sigma)$. The objective of the graphing algorithm is to extract the locus, $s(\sigma)$, for every feature.

5.2.2 Algorithm Design

The samples or nodes of the scale space feature sketch can be found by applying 2D non-maximum suppression to $D(\mathbf{x}, \sigma)$ for all σ. This produces the nodes of the graph, but does not provide the graph structure. The graphing algorithm assembles the feature sample points into graph structures that represent the sketch.

The graphing algorithm and graph data structure are derived from the properties and behaviour of the extrema of saliency functions in scale space. These properties are enumerated in Chapter 4, Sections 4.1.2 and 4.1.3. The properties that play a primary role in the determining the rules of the algorithm are summarised as follows,

1. Extrema may merge, but may not diverge as scale increases.
2. Extrema may be created or annihilated.
3. Extrema move by a finite amount over a finite change in scale.
4. Extrema generated by different structures are, by definition, separated by edges in the image. They will be separated by a distance related to the scale and will diverge.
5. Extrema generated by closely related structures tend to converge.
6. The velocity of extrema over scale can only be predicted through extensive analysis of the image gradients.

From the first property it is clear that the feature sketch graph will take the form of a forest of tree graphs. The root of a tree in the graph is the highest scale feature in that tree. Connected elements in the tree are at adjacent scales. The data structures chosen to represent the graph make use of a bottom-up approach in terms of scale. The graph is represented by the vector of features, f, and two index vectors, p and l. The feature vector, f, lists the scale space coordinates of each feature. Each element in index vector p corresponds to an element in the features vector, f, and lists the index of the feature's parent feature – the parent of feature $f\left(i\right)$ is $f\left(p\left(i\right)\right)$. Any feature's parent is at a higher scale than the feature. In the case of root nodes (that do not have higher scale parents), $p\left(i\right) = i$. Index vector l lists the indexes of the leaf nodes. This representation encodes convergences implicitly, since multiple nodes may have the same parent.

The graph is constructed by a simple bottom-up method. The parent of each feature in a given scale level is found by searching the next scale level up, in the spatial vicinity of the feature. The closest feature found within a set radius is chosen as the parent. Note that direct connectivity (such as eight-connectivity) is not required. Because features from different structures tend to diverge and features from within a closed structure tend to converge, the most appropriate parent of any feature is the one closest to it in the next scale. It is possible that a parent may be found when in fact the feature has been annihilated, though this is sufficiently unlikely. The nearest feature search strategy is chosen because predicting the displacement of a feature over scale requires analysing the gradients in the vicinity of the feature, which is computationally demanding. Features for which no suitable parent can be found are root nodes. Features that are not the parent of any other node are leaf nodes (lowest scale node in a branch) and are listed in l.

The size of the area searched for the parent of a feature must be large enough to include all features from the same structure. At the same time, features from other structures should be avoided. When a feature is annihilated, the highest scale feature before the annihilation is the root feature of its tree. A maximum search radius must be chosen carefully to avoid incorporating such trees into nearby trees, instead of creating the appropriate root node. The velocity over scale of features located near corners depends on the angle of the corner and can be very large where the angle is very small. Converging features accelerate as they approach convergence. The situation is exaggerated if the scale-space is sampled very sparsely, so that there is a large scale change between successive levels. Based on these observations, it is desirable to maximise the search radius, to improve the likelihood of locating features that belong to the same structure.

The dominant limiting factor in choosing a search radius is the minimum distance that features from different structures can be located from each other. The determinant of Hessian operator is considered as an example. This operator produces maxima at a distance of σ from an edge. Therefore, features from opposite sides of an edge will be located 2σ or more from each other at any particular scale, until other image structures start to affect the feature position. The search radius limit for determinant of Hessian features is chosen as $1.5\sigma_i$ (where σ_i is the scale of the feature for which a parent is sought) to compromise between the search area for a given feature and the search area of a potential neighbouring feature at the same scale. This should only be reduced if the pyramid is sampled very densely. The graphing algorithm accepts the parameter k, and sets the search radius to $k\sigma$, where k should be chosen based on the specific behaviour of the saliency operator used.

5.2.3 The Algorithm

The scale space feature sketch graph construction algorithm is listed in Algorithm 5.1. Lines 5.1.7 - 5.1.9 build the list of leaf nodes. Lines 5.1.10 - 5.1.11 search for the parent of feature f_i. The function, $t \leftarrow \text{FINDNEAR}(\mathbf{x}, \sigma, r)$, is discussed in detail below. If a suitable parent is found, its index is recorded in

Algorithm 5.1: The Scale Space Feature Sketch Graph Construction Algorithm.

Function:
$\{p, l\} \leftarrow \text{GRAPH}(f, k)$.
Input:
$f = \{\{\mathbf{x}, \sigma\}_1, \{\mathbf{x}, \sigma\}_2, \ldots, \{\mathbf{x}, \sigma\}_n\}$ – A set of n feature coordinates in discrete scale space with scales $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$, sorted according to ascending scale.
k – The search radius multiplier.
Output:
$p = \{p_1, p_2, \ldots, p_n\}$ – The output graph. Each element, p_i, corresponds to a feature f_i at the same index and lists the index of the parent of f_i. Root nodes list themselves as parent.
$l = \{l_1, l_2, \ldots, l_o\}$ – The list of leaf node indexes.

```
5.1.1  begin
5.1.2      $p \leftarrow \{0, 0, \ldots, 0\}$ ($n$ zeros).
5.1.3      $l \leftarrow \emptyset$.
5.1.4      $i \leftarrow 1$.
5.1.5      repeat
5.1.6          $f_i = \{\mathbf{x}_i, \sigma_j\}$.
5.1.7          if $p_i = 0$ then
5.1.8              $l \leftarrow \{l, i\}$.
5.1.9          end
5.1.10         $r \leftarrow k\sigma_j$.
5.1.11         $t \leftarrow \text{FINDNEAR}(\mathbf{x}_i, \sigma_{j+1}, r)$.
5.1.12         if $t = 0$ then
5.1.13             $p_i \leftarrow i$.
5.1.14         else
5.1.15             $p_i \leftarrow t$.
5.1.16             $p_t \leftarrow 1$.
5.1.17         end
5.1.18         $i \leftarrow i + 1$.
5.1.19     until $\sigma_j = \sigma_m$
5.1.20     while $i \leq n$ do
5.1.21         if $p_i = 0$ then
5.1.22             $l \leftarrow \{l, i\}$.
5.1.23         end
5.1.24         $p_i \leftarrow i$.
5.1.25         $i \leftarrow i + 1$.
5.1.26     end
5.1.27 end
```

the appropriate element in p, line 5.1.15, otherwise, the node is marked as a root node, line 5.1.13. Line 5.1.16 ensures that nodes that have children are not enlisted as leaf nodes. Features at the highest scale level, processed in lines 5.1.20 - 5.1.26, do not have parents, and therefore do not require the parent search step. They are all labelled as parents and those that are leaf nodes are appended to the list of leaf nodes.

The function, $t \leftarrow \text{FINDNEAR}(\mathbf{x}, \sigma, r)$, searches the set of input features for features at scale level σ that are within radius r of the spatial coordinates $\mathbf{x}$. The function returns the index of the feature closest to $\mathbf{x}$. If no feature is found within the specified radius, then zero is returned. The implementation of this function should be optimised based on the arrangement of the feature set. For example, if the non-maximum suppression operation that was used to generate the feature set was run over one scale level at a time from bottom to top in the pyramid and from top to bottom in the image, then the features are sorted according to scale level and each level is sorted according to the y coordinate. While searching for the parents of features at scale σ_j, a search window can be maintained at scale σ_{j+1} over a y coordinate range of r. As processing continues, the y coordinate of the features being processed will increase monotonously and the search window is updated accordingly. In this way the number of features searched is limited to only a few features, instead of the whole feature set.

Figures 5.1 and 5.2 show visualisations of the scale space feature sketch graphs superimposed on a set of multi-scale determinant of Hessian features. The effect of scale sampling density is shown in Figure 5.3. A consistent sketch is produced for various scale sampling densities. A sketch produced from a lower scale sampling density feature set is approximately equivalent to one produced from a higher density set, but with detail removed. At very low sampling densities, errors become more common. Figure 5.3 shows an example where sampling one level

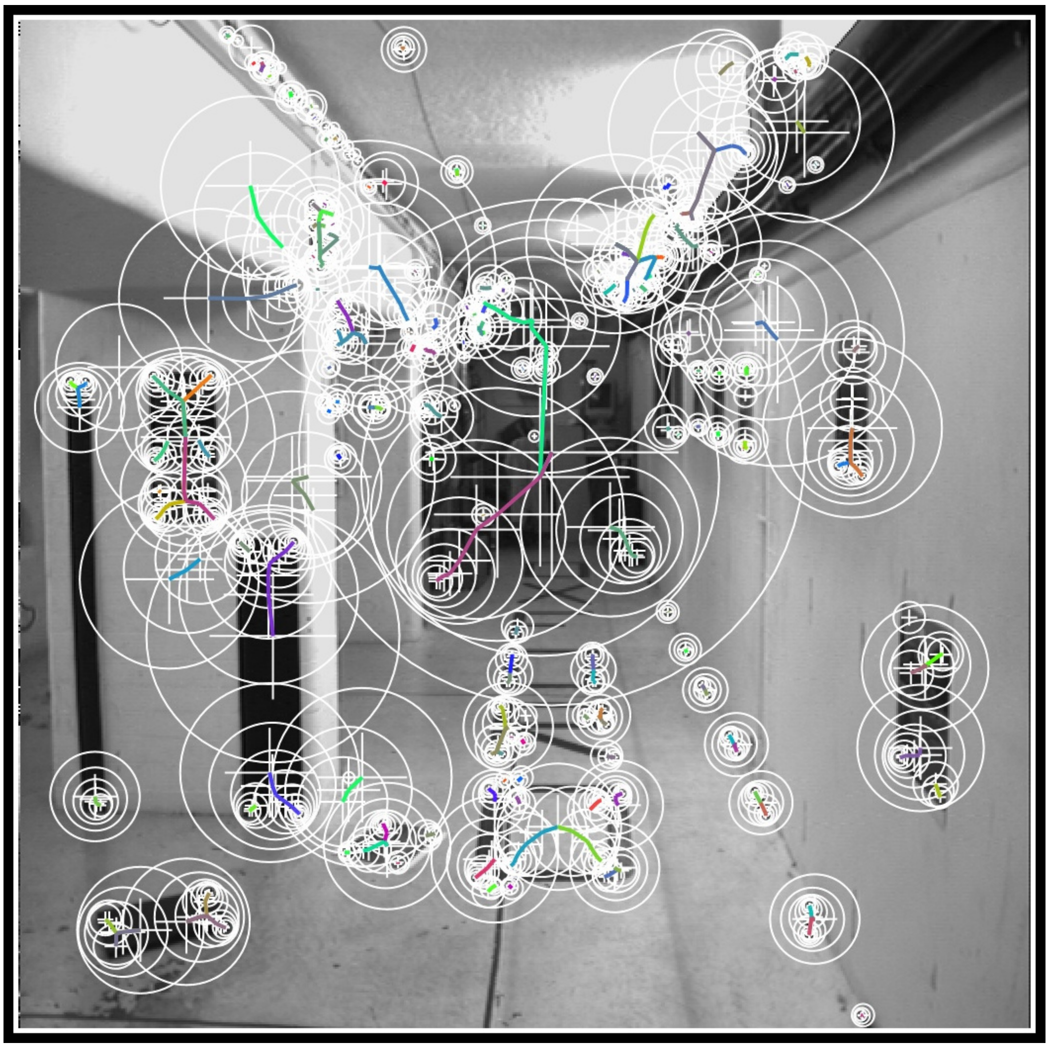


Figure 5.1: Output of scale space feature sketch graphing algorithm when applied to Determinant of Hessian features. Each white circle with a cross at its centre represents a feature. The radius of the circle indicates the feature scale. The scale space feature sketch is indicated by coloured lines joining the centres of the feature circles. Each branch of the sketch was assigned a random colour.

per scale octave results in some of the branches of the graph not being connected to the nearby tree, in contrast to the higher sampling density cases, where these branches are connected.

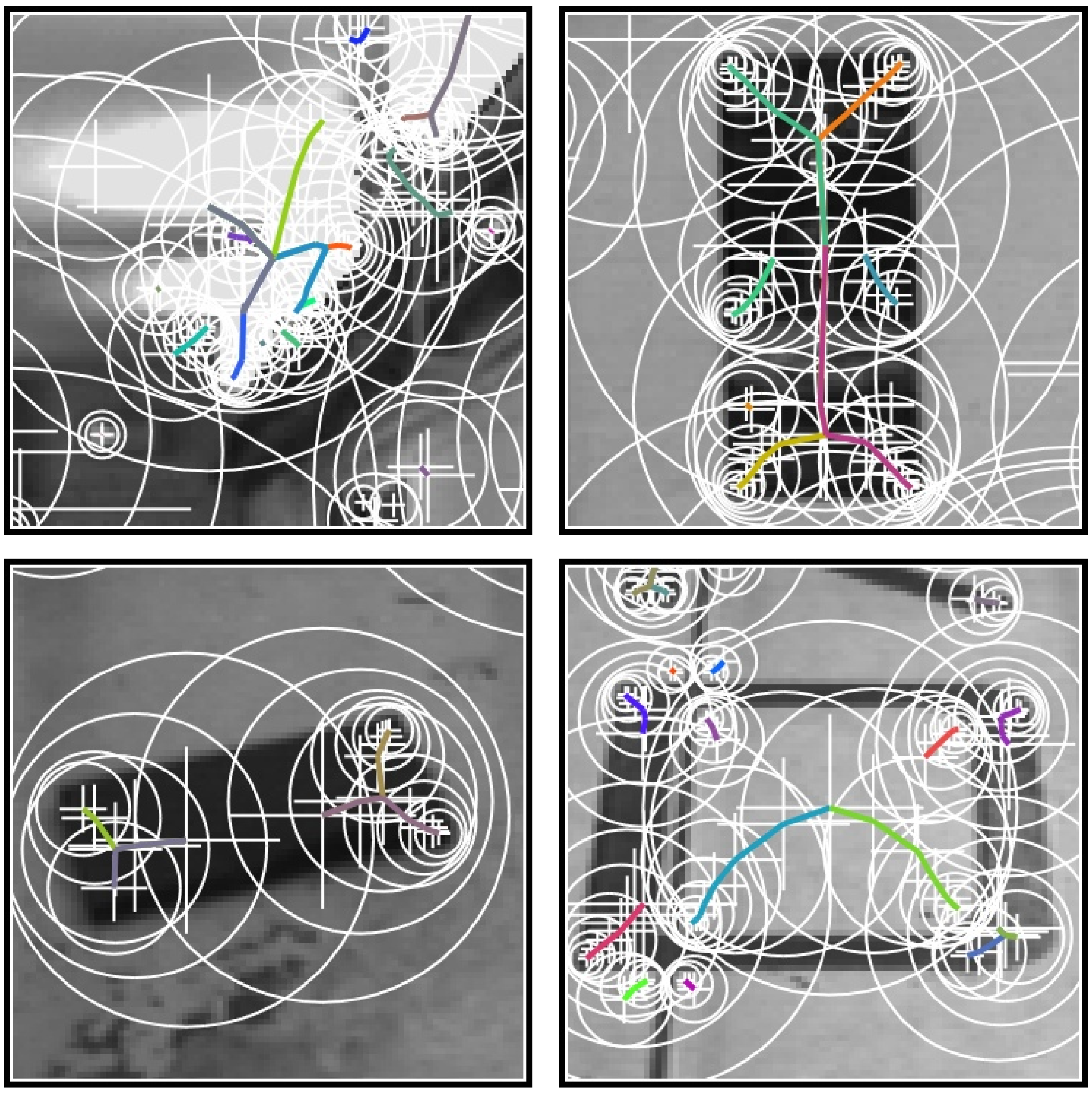


Figure 5.2: Enlarged subregions of Figure 5.1

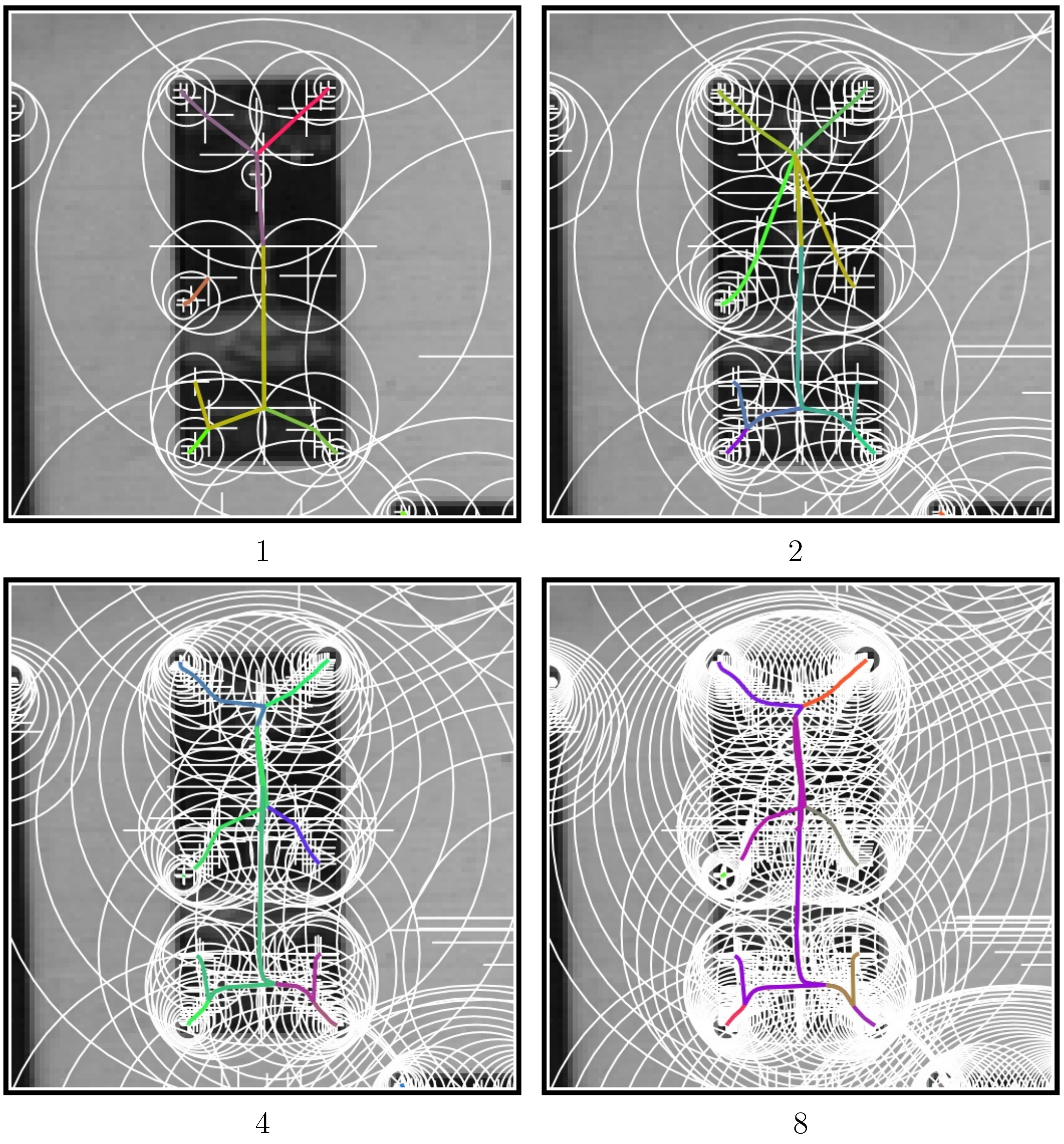


Figure 5.3: Output of scale space feature sketch graphing algorithm for four different scale sampling densities – one, two, four and eight samples per scale octave, as indicated.

5.3 Characteristic Scale Selection using Scale Space Primal Sketch Graphs

The characteristic scale features in a primal sketch may be identified using a very simple procedure. First some scale response function is evaluated at each feature's coordinates in the image scale space. This could be the saliency function, the Laplacian, or some other function of scale space derivatives. Characteristic features are then selected as the features that yield a response higher than their neighbours in the graph.

Algorithm 5.2 presents an algorithm for performing characteristic scale feature selection. At each point in the graph, the responses of three nodes along the branch are compared to check for the presence of a local maximum in the second node (lines 5.2.4 - 5.2.14). If a maximum is found, it is marked in a label array, k. At the end, the label array is converted to array of indexes of selected features (line 5.2.15).

5.3.1 Evaluation

In this section the scale space feature sketch (SSFS) based scale selection method is compared to the scale space 3D maxima method. The testing method described in [45] and Chapter 3, Section 3.3.1 is used.

A set of feature extractors were set up that used different saliency operators, scale response operators, scale selection methods and affine adaptation methods. Each extractor was given a shorthand name based on its configuration. The saliency operators tested were the Determinant of Hessian operator (labelled He) and the Harris operator (labelled Ha). The scale response operator was either the saliency

Algorithm 5.2: The Scale Space Primal Sketch-based Scale Selection Algorithm.

Function:
$s \leftarrow \mathrm{SELECT}\left(f, p, R\left(\mathbf{x}, \sigma\right)\right)$.
Input:
$f = \left\{\left\{\mathbf{x}, \sigma\right\}_1, \left\{\mathbf{x}, \sigma\right\}_2, \ldots, \left\{\mathbf{x}, \sigma\right\}_n\right\}$ – A set of n feature coordinates in discrete scale space.
$p = \left\{p_1, p_2, \ldots, p_n\right\}$ – The feature sketch graph. Each element, p_i, corresponds to a feature f_i at the same index and lists the index of the parent of f_i. Root nodes list themselves as parent.
$R\left(\mathbf{x}, \sigma\right)$ – The scale response function (may be pre-computed at each feature location).
Output:
$s = \left\{s_1, s_2, \ldots, s_m\right\}$ – The indexes of the selected features.

```
5.2.1  begin
5.2.2      $i_1 \leftarrow 1$.
5.2.3      $k \leftarrow \{0, 0, \ldots, 0\}$ ($n$ zeros, Boolean).
5.2.4      while $i_1 \leq n$ do
5.2.5          $i_2 \leftarrow p(i_1)$.
5.2.6          $i_3 \leftarrow p(i_2)$.
5.2.7          $f_1 = f(i_1)$.
5.2.8          $f_2 = f(i_2)$.
5.2.9          $f_3 = f(i_3)$.
5.2.10         $r_1 \leftarrow R(\mathbf{x}_1, \sigma_1)$.
5.2.11         $r_2 \leftarrow R(\mathbf{x}_2, \sigma_2)$.
5.2.12         $r_3 \leftarrow R(\mathbf{x}_3, \sigma_3)$.
5.2.13         $k_i \leftarrow (r_2 > r_1) \cdot (r_2 > r_3)$.
5.2.14     end
5.2.15     $s \leftarrow \arg(k \neq 0)$.
5.2.16 end
```

operator or the Laplacian operator. Extractors using the Laplacian operator as scale response are given subscript l. Two scale selection methods were tested – the scale space feature sketch-based method (given subscript P) and the 3D non-maximum suppression method (given subscript 3). Iterative scale selection is not included in the evaluation because it does not provide significant benefits over the other method, but requires much greater computation time. Extractors were tested with and without affine adaptation. Those that employ affine adaptation are given subscript A. This gives a total of eight extractors tested.

All tests used a Gaussian scale space pyramid computed over the same scale range with the same total sub-sampling factor, and the same number of scale levels per octave (see Chapter 4 for a discussion on scale space). The 3D non maximum suppression methods used pyramids in which the sampling remained the same for an octave, and each successive octave was down-sampled. Additional levels were computed at the start and end of each octave to facilitate 3D operations in the first and last levels. The primal sketch-based methods used pyramids in which each successive level was down-sampled. Both pyramids down-sampled by a factor of $\sqrt{2}$ per octave. The SIFT descriptor [38, 39] was used for all extractors.

Results

The analysis of test results is listed in Tables 5.1 -5.16. Tables 5.1 - 5.8 present paired T-tests comparing the SSFS-based methods to the 3D methods. Tables 5.9 - 5.16 present paired T-tests comparing the methods using the saliency function as scale response to the methods that use the Laplacian.

Figure 5.4 presents a visual summary of the relative performance of the two scale selection methods. Box plots of the log of the ratio of test scores of the two methods are shown.

	mean(He_P)	mean(He_3)	mean(He_P / He_3)	p
repeatability (%)	61.14	61.12	1.01	0.98
correspondences	1579.67	1569.00	0.98	0.66
matching score (%)	21.87	21.22	1.04	0.01
correct matches	508.07	506.03	1.01	0.78
efficiency (n/s)	922.88	709.40	1.30	0.00

Table 5.1: Paired T-test comparing results of He_P and He_3.

	mean(He_{lP})	mean(He_{l3})	mean(He_{lP} / He_{l3})	p
repeatability (%)	60.48	54.03	1.14	0.00
correspondences	1569.67	1527.87	1.00	0.45
matching score (%)	21.06	14.20	1.49	0.00
correct matches	496.47	376.20	1.31	0.00
efficiency (n/s)	873.95	617.27	1.40	0.00

Table 5.2: Paired T-test comparing results of He_{lP} and He_{l3}.

	mean(He_{PA})	mean(He_{3A})	mean(He_{PA} / He_{3A})	p
repeatability (%)	56.04	57.57	0.98	0.01
correspondences	1192.30	1219.40	0.95	0.07
matching score (%)	20.86	21.33	0.99	0.08
correct matches	393.90	411.57	0.96	0.00
efficiency (n/s)	100.01	26.57	4.81	0.00

Table 5.3: Paired T-test comparing results of He_{PA} and He_{3A}.

	mean(He_{lPA})	mean(He_{l3A})	mean(He_{lPA} / He_{l3A})	p
repeatability (%)	54.99	49.63	1.12	0.00
correspondences	1154.00	1153.43	0.97	0.99
matching score (%)	20.42	13.15	1.55	0.00
correct matches	382.10	285.03	1.34	0.00
efficiency (n/s)	97.74	21.84	4.40	0.00

Table 5.4: Paired T-test comparing results of He_{lPA} and He_{l3A}.

	mean(Ha_P)	mean(Ha_3)	mean(Ha_P / Ha_3)	p
repeatability (%)	51.38	52.68	1.02	0.71
correspondences	1028.33	396.97	2.26	0.00
matching score (%)	18.22	21.84	0.88	0.01
correct matches	314.17	137.73	1.93	0.00
efficiency (n/s)	197.82	57.18	3.04	0.00

Table 5.5: Paired T-test comparing results of Ha_P and Ha_3.

	mean(Ha_{lP})	mean(Ha_{l3})	mean(Ha_{lP} / Ha_{l3})	p
repeatability (%)	54.43	45.05	1.23	0.00
correspondences	1241.80	474.93	2.83	0.00
matching score (%)	18.14	15.07	1.26	0.00
correct matches	361.77	130.43	2.81	0.00
efficiency (n/s)	230.56	77.09	2.99	0.00

Table 5.6: Paired T-test comparing results of Ha_{lP} and Ha_{l3}.

	mean(Ha_{PA})	mean(Ha_{3A})	mean(Ha_{PA} / Ha_{3A})	p
repeatability (%)	44.56	50.48	0.93	0.07
correspondences	708.90	332.47	1.80	0.00
matching score (%)	19.38	22.48	1.02	0.01
correct matches	253.63	132.07	1.86	0.00
efficiency (n/s)	18.35	2.72	17.55	0.00

Table 5.7: Paired T-test comparing results of Ha_{PA} and Ha_{3A}.

	mean(Ha_{lPA})	mean(Ha_{l3A})	mean(Ha_{lPA} / Ha_{l3A})	p
repeatability (%)	46.56	40.69	1.21	0.00
correspondences	750.07	380.40	2.03	0.00
matching score (%)	17.92	13.75	1.36	0.00
correct matches	245.50	117.20	2.25	0.00
efficiency (n/s)	13.02	2.51	8.27	0.00

Table 5.8: Paired T-test comparing results of Ha_{lPA} and Ha_{l3A}.

	mean(He_{l3})	mean(He_{3})	mean(He_{l3} / He_{3})	p
repeatability (%)	54.03	61.12	0.88	0.00
correspondences	1527.87	1569.00	0.99	0.38
matching score (%)	14.20	21.22	0.69	0.00
correct matches	376.20	506.03	0.77	0.00
efficiency (n/s)	617.27	709.40	0.89	0.00

Table 5.9: Paired T-test comparing results of He_{l3} and He_{3}.

	mean(He_{l3A})	mean(He_{3A})	mean(He_{l3A} / He_{3A})	p
repeatability (%)	49.63	57.57	0.86	0.00
correspondences	1153.43	1219.40	0.98	0.05
matching score (%)	13.15	21.33	0.65	0.00
correct matches	285.03	411.57	0.73	0.00
efficiency (n/s)	21.84	26.57	1.00	0.09

Table 5.10: Paired T-test comparing results of He_{l3A} and He_{3A}.

	mean(Ha_{l3})	mean(Ha_3)	mean(Ha_{l3} / Ha_3)	p
repeatability (%)	45.05	52.68	0.88	0.00
correspondences	474.93	396.97	1.13	0.04
matching score (%)	15.07	21.84	0.80	0.00
correct matches	130.43	137.73	0.96	0.36
efficiency (n/s)	77.09	57.18	1.41	0.00

Table 5.11: Paired T-test comparing results of Ha_{l3} and Ha_3.

	mean(Ha_{l3A})	mean(Ha_{3A})	mean(Ha_{l3A} / Ha_{3A})	p
repeatability (%)	40.69	50.48	0.84	0.00
correspondences	380.40	332.47	1.08	0.09
matching score (%)	13.75	22.48	0.82	0.00
correct matches	117.20	132.07	1.01	0.02
efficiency (n/s)	2.51	2.72	1.49	0.67

Table 5.12: Paired T-test comparing results of Ha_{l3A} and Ha_{3A}.

	mean(He_{lP})	mean(He_P)	mean(He_{lP} / He_P)	p
repeatability (%)	60.48	61.14	0.99	0.02
correspondences	1569.67	1579.67	1.00	0.41
matching score (%)	21.06	21.87	0.97	0.00
correct matches	496.47	508.07	0.97	0.02
efficiency (n/s)	873.95	922.88	0.93	0.00

Table 5.13: Paired T-test comparing results of He_{lP} and He_P.

	mean(He_{lPA})	mean(He_{PA})	mean(He_{lPA} / He_{PA})	p
repeatability (%)	54.99	56.04	0.98	0.00
correspondences	1154.00	1192.30	0.98	0.00
matching score (%)	20.42	20.86	0.98	0.04
correct matches	382.10	393.90	0.98	0.01
efficiency (n/s)	97.74	100.01	0.99	0.02

Table 5.14: Paired T-test comparing results of He_{lPA} and He_{PA}.

	mean(Ha_{lP})	mean(Ha_P)	mean(Ha_{lP} / Ha_P)	p
repeatability (%)	54.43	51.38	1.21	0.18
correspondences	1241.80	1028.33	1.60	0.00
matching score (%)	18.14	18.22	1.23	0.94
correct matches	361.77	314.17	1.62	0.00
efficiency (n/s)	230.56	197.82	1.60	0.00

Table 5.15: Paired T-test comparing results of Ha_{lP} and Ha_P.

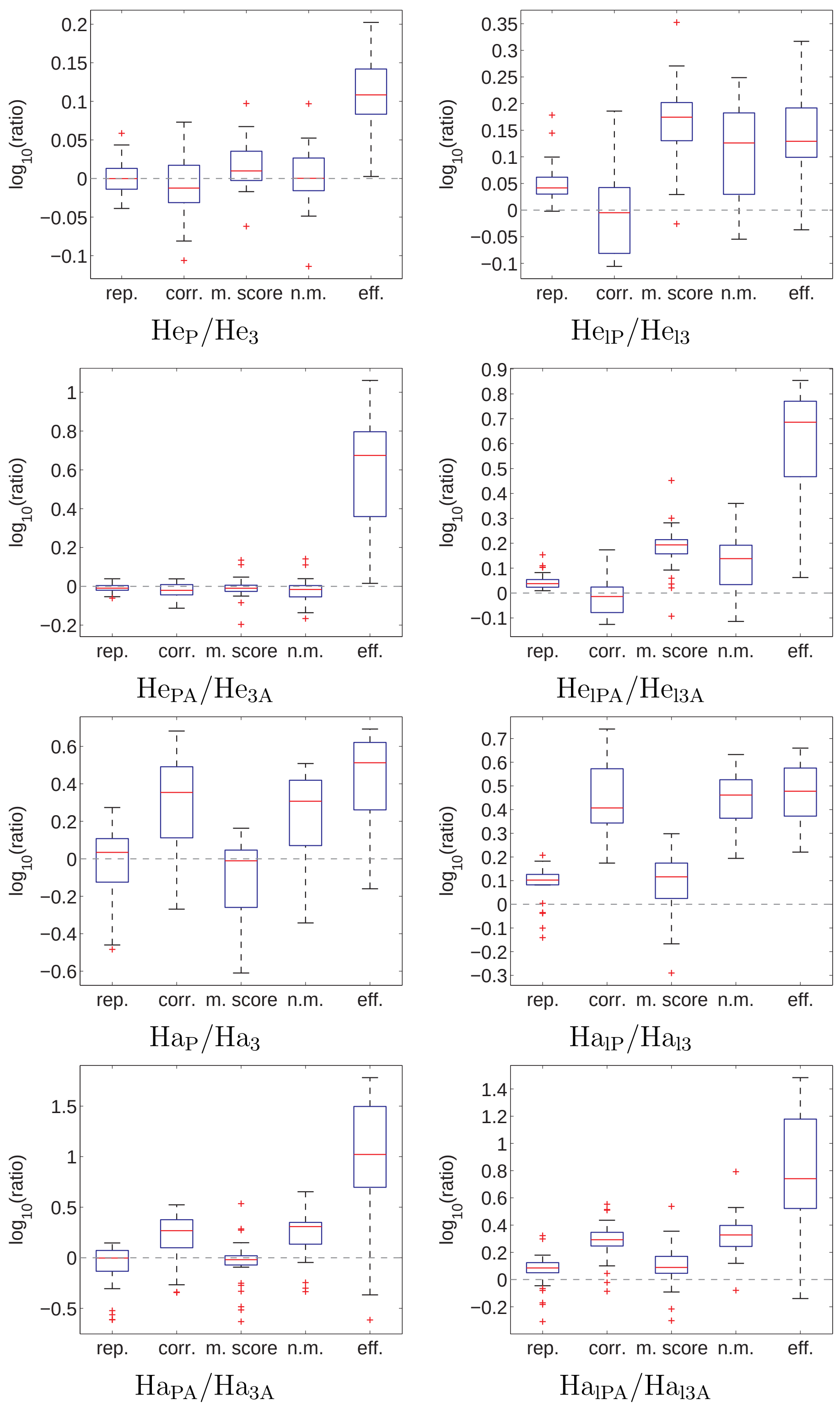


Figure 5.4: Box plots of the log of the ratio of extractors using the scale space feature sketch scale selection method and extractors using 3D non-maximum suppression.

	mean(Ha_{lPA})	mean(Ha_{PA})	mean(Ha_{lPA} / Ha_{PA})	p
repeatability (%)	46.56	44.56	1.23	0.28
correspondences	750.07	708.90	1.40	0.06
matching score (%)	17.92	19.38	1.13	0.20
correct matches	245.50	253.63	1.28	0.44
efficiency (n/s)	13.02	18.35	0.95	0.00

Table 5.16: Paired T-test comparing results of Ha_{lPA} and Ha_{PA}.

Discussion

The SSFS-based scale selection method achieves superior computational efficiency in all tests. When using the Determinant of Hessian extractor and no affine adaptation, SSFS achieves a small gain of 30% to 40% on average (Tables 5.1 - 5.2). With the Harris detector, SSFS is three times as efficient as the 3D method (Tables 5.5 - 5.6).

For the extractors that make use of affine adaptation, the SSFS-based methods yield greatly superior efficiency compared to the 3D-based methods. On average a 4.4 to 4.8 fold increase in efficiency is observed for the Determinant of Hessian-based extractors (Tables 5.3 - 5.4) and 8.3 to 17.6 times increase is observed for the Harris-based extractors (Tables 5.7 - 5.8). The plots in Figure 5.4 show an efficiency increase in excess of ten fold for a significant proportion of trials. The same extractors do not show an increase in correct match counts of comparable magnitude to the increase in efficiency. This suggests that the SSFS-based scale selection method produces features that complete the affine adaptation process in fewer iterations.

The two scale selection techniques yield similar performance when using the Determinant of Hessian extractor (Table 5.1). When using the Harris extractor, the SSFS-based method consistently produces at least twice as many correspondences and matches, and three times higher efficiency. The matching score is reduced by a factor 0.88 (Table 5.5).

When using the Laplacian as scale response function, the SSFS-based method consistently achieves higher repeatability (10% to 20% increase), matching score (approximately 50% increase) and number of correct matches (30% to 40% increase), compared with the 3D method (Tables 5.2, 5.4, 5.8, 5.8). Tables 5.9 - 5.16 compare the methods using the saliency function as scale response to the methods that use the Laplacian. When comparing the performance of the Laplacian and the saliency function as scale response functions, the 3D-based methods show a degradation in performance in all metrics apart from the number of correspondences (Tables 5.9 - 5.12). This degradation is most significant when using the Determinant of Hessian extractor. The matching score metric is most strongly affected with as much as 35% reduction in score. The SSFS-based methods do not show the same degradation in performance (no more than 5% in any metric, Tables 5.13 - 5.16). For the Determinant of Hessian extractor using SSFS-based scale selection, there is no noticeable difference between using the Laplacian and the saliency function as scale response function. This is expected, since the Determinant of Hessian and Laplacian are very similar functions. For the Harris extractor, using the Laplacian results in a 60% increase in correspondences, correct matches and efficiency. These results support the hypothesis that the SSFS-based scale selection is more effective than the 3D method, when using the Laplacian as scale response function.

Overall, the SSFS-based scale selection method achieves results that are equivalent or superior to the 3D non-maximum suppression method in all tests. The SSFS-based method is the better choice, by a convincing margin, when using the Harris extractor, the Laplacian scale response function, or affine adaptation.

5.4 Chapter Summary

In this chapter, the scale space primal sketch is introduced as a tool for multi-scale analysis and scale selection. The scale space feature sketch is proposed – a primal sketch where the primitives are salient features. This is a simple extension and does not require any modification of the scale space primal sketch concept.

An algorithm is presented for computing the SSFS from a set of multi-scale features. The design of this algorithm is based on the knowledge of how saliency map-based features typically behave in scale space. It requires no further image processing, operating only on the set of feature coordinates.

A novel method for selecting characteristic scale features from the SSFS is presented. The new method is compared experimentally with 3D scale space non-maximum suppression. Using the SSFS for scale selection has four principal advantages over the existing 3D scale space non-maximum suppression method: Firstly, it does not impose restrictions on how the scale space is sampled. Secondly, greater accuracy is achieved through the ability to accurately track features over scale. Thirdly, the scale response function can be evaluated strictly along the feature locus, making it possible to select the best feature. This gives better scale selection results than somewhat arbitrarily asserting that the scale response of a feature must assume a local maximum over scale. This property is especially beneficial when different functions are used for selecting spatial position and scale position. Finally, greater computational efficiency is achieved, especially where affine adaptation is applied subsequent to scale selection.

Chapter 6

Affine Adaptation using the Hessian Matrix

The second moment matrix is currently the dominant affine shape estimator used in affine adaptation of saliency map-based local image features. This chapter explores the novel approach of using the Hessian matrix as an affine shape measure for affine adaptation. Both operators are introduced in Chapter 4, Section 4.2.1. Affine adaptation using the second moment matrix is reviewed in Section 4.3.2. The main motivations for using the Hessian matrix instead of the second moment matrix are that the Hessian is simpler to implement and requires less computational effort. The experiments in Section 6.5 show that it is also more effective in practical problems. The limitations associated with the Hessian matrix are also explored.

The ability of the Hessian matrix to measure affine shape is demonstrated from two points of view. Section 6.1 shows how the Hessian matrix can be used to measure the covariance matrix of a Gaussian blob up to scale. In Section 6.2 isotropy is defined in terms of the Hessian itself. It is shown that the Hessian can be used to measure the second-order shape of structures in scale space with arbitrary shape. In both cases the model is an approximation to the real image feature shape. The process must be applied iteratively in order to find the true affine shape of a feature.

Implementation issues are briefly considered in Section 6.3. Section 6.4 compares the complexity of the Hessian and second moment operators. The performance of the Hessian matrix and second moment matrix used as affine shape estimators is compared in Section 6.5. The chapter is concluded in Section 6.6.

6.1 The Affine Gaussian Model

In this section it is demonstrated that the Hessian matrix can be used to measure the covariance matrix of an affine Gaussian function up to scale. This is then applied to real image features in scale space.

The 2D affine Gaussian function centred on the coordinate origin is defined as,

$$g(\mathbf{x}, \Sigma) = \frac{1}{2\pi |\mathbf{\Sigma}|} e^{\frac{-\mathbf{x}^\top \mathbf{\Sigma}^{-1} \mathbf{x}}{2}}.$$

The matrix $\mathbf{\Sigma}$ may be decomposed as,

$$\mathbf{\Sigma} = \sigma_\alpha^2 \mathbf{\Sigma}' = \sigma_\alpha^2 \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix}, \tag{6.1}$$

so that $\det(\mathbf{\Sigma}) = \sigma_\alpha^4$, $\det(\mathbf{\Sigma}') = 1$, $\sigma_{xx} > 0$ and $\sigma_{yy} > 0$. Rewriting $g(\mathbf{x}, \mathbf{\Sigma})$ with arbitrary gain k in terms of $\mathbf{\Sigma}'$ and σ_α yields,

$$g(\mathbf{x}, \mathbf{\Sigma}) = k e^{\frac{-\mathbf{x}^\top \mathbf{\Sigma}'^{-1} \mathbf{x}}{2\sigma_\alpha^2}}.$$

The second order partial derivatives (the Hessian) of $g(\mathbf{x}, \mathbf{\Sigma})$ are,

$$\begin{aligned}
\tfrac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial x^2} &= kg(\mathbf{x},\mathbf{\Sigma})\left(\tfrac{1}{\sigma_\alpha^4}\left(x^2\sigma_{\mathrm{yy}}^2 - 2xy\sigma_{\mathrm{yy}}\sigma_{\mathrm{xy}} + y^2\sigma_{\mathrm{xy}}^2\right) - \tfrac{\sigma_{\mathrm{yy}}}{\sigma_\alpha^2}\right), \\
\tfrac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial y^2} &= g(\mathbf{x},\mathbf{\Sigma})\left(\tfrac{1}{\sigma_\alpha^4}\left(x^2\sigma_{\mathrm{xy}}^2 - 2xy\sigma_{\mathrm{xy}}\sigma_{\mathrm{xx}} + y^2\sigma_{\mathrm{xx}}^2\right) - \tfrac{\sigma_{\mathrm{xx}}}{\sigma_\alpha^2}\right), \\
\tfrac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial x \partial y} &= g(\mathbf{x},\mathbf{\Sigma})\left(\tfrac{1}{\sigma_\alpha^4}\left(-x^2\sigma_{\mathrm{xy}}\sigma_{\mathrm{yy}} + xy\left(\sigma_{\mathrm{xx}}\sigma_{\mathrm{yy}} + \sigma_{\mathrm{xy}}^2\right) - y^2\sigma_{\mathrm{xx}}\sigma_{\mathrm{xy}}\right) + \tfrac{\sigma_{\mathrm{xy}}}{\sigma_\alpha^2}\right), \\
\tfrac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial y \partial x} &= \tfrac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial x \partial y}.
\end{aligned}$$

In the following discussion the second order partial derivatives of g will be indicated as,

$$\begin{aligned}
g_{xx}(\mathbf{x},\mathbf{\Sigma}) &= \tfrac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial x^2}, \\
g_{yy}(\mathbf{x},\mathbf{\Sigma}) &= \tfrac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial y^2}, \\
g_{xy}(\mathbf{x},\mathbf{\Sigma}) &= \tfrac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial x \partial y} = \tfrac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial y \partial x}.
\end{aligned}$$

Evaluating the second order partial derivatives at $\mathbf{x} = \mathbf{0}$ gives,

$$\begin{aligned}
g_{xx}(\mathbf{0},\mathbf{\Sigma}) &= k\tfrac{-\sigma_{yy}}{\sigma_\alpha^2} &= -k'\sigma_{yy}, \\
g_{yy}(\mathbf{0},\mathbf{\Sigma}) &= k\tfrac{-\sigma_{xx}}{\sigma_\alpha^2} &= -k'\sigma_{xx}, \\
g_{xy}(\mathbf{0},\mathbf{\Sigma}) &= k\tfrac{\sigma_{xy}}{\sigma_\alpha^2} &= k'\sigma_{xy}.
\end{aligned}$$

In matrix form, the Hessian of a Gaussian evaluated at the Gaussian centre is a scaled version of the negative inverse of the Gaussian's covariance matrix,

$$\left.\frac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial \mathbf{x}^2}\right|_{\mathbf{x}=\mathbf{0}} = -k'\begin{bmatrix} \sigma_{yy} & -\sigma_{xy} \\ -\sigma_{xy} & \sigma_{xx} \end{bmatrix} = -k'\mathbf{\Sigma}^{-1}.$$

Negating the sign of the Gaussian negates the sign of the Hessian.

Hence $\mathbf{\Sigma}'$ can be recovered by evaluating the inverse Hessian matrix at the centre of the Gaussian function,

$$\begin{aligned}
\mathcal{H}' &= \left.\frac{\partial^2 g(\mathbf{x},\mathbf{\Sigma})}{\partial \mathbf{x}^2}\right|_{\mathbf{x}=\mathbf{0}}, \\
\mathcal{H} &= \frac{\mathcal{H}'\operatorname{sign}\left(\mathcal{H}'_{0,0}\right)}{\sqrt{\det(\mathcal{H}')}}, \\
\mathbf{\Sigma}' &= \mathcal{H}^{-1},
\end{aligned}$$

where $\text{sign}\big(\mathcal{H}'_{0,0}\big)$ gives the sign ($+1$ or -1) of the top left element of $\mathcal{H}'$. To normalise the anisotropic blob to an isotropic blob, simply transform the image by the square root of $\mathbf{\Sigma}'$,

$$\begin{aligned} g\left(\mathbf{\Sigma}'^{\frac{1}{2}}\mathbf{x}, \mathbf{\Sigma}\right) &= ke^{\frac{-\mathbf{x}^\top \mathbf{\Sigma}'^{\frac{1}{2}\top} \mathbf{\Sigma}'^{-1} \mathbf{\Sigma}'^{\frac{1}{2}} \mathbf{x}}{2\sigma_\alpha^2}} \\ &= ke^{\frac{-\mathbf{x}^\top \mathbf{x}}{2\sigma_\alpha^2}}. \end{aligned}$$

This proof shows how to directly measure the affine shape of a Gaussian function of which the centre point is known. This method may be applied to local image features, despite the fact that image structures are, in general, not proper affine Gaussian functions. The characteristic scale representations of blob-like structures are sufficiently close to affine Gaussian blobs since they have been convolved with a Gaussian of the same scale as the structure. A characteristic scale blob extractor, such as the Laplacian or determinant of Hessian extractor, can be used to find the blob scale and centre point.

The Hessian matrix can therefore be used for affine normalisation in conjunction with a characteristic scale blob extractor. This system does not, however, provide a direct solution. Convolving an isotropic function with an isotropic Gaussian produces another isotropic function, but convolving an anisotropic function with a Gaussian produces a function with a different shape to the original. The Hessian therefore only provides an approximate shape measure when applied to a characteristic scale shape.

An iterative technique can be used to achieve an accurate shape measure. At each iteration, the shape is measured with the Hessian and then normalised by transforming the shape with the inverse square root of the normalised Hessian. After each iteration the normalised shape is closer to isotropic, resulting in a more accurate shape estimate during the following iteration. This continues until the shape is measured to be sufficiently close to isotropic. The normalisation transformation accumulated over all iterations represents the final affine covariant shape measure.

6.2 Isotropic Local Hessian

In the previous section, the concept of isotropy was defined in terms of the covariance matrix of a Gaussian blob. In this section, isotropy is instead defined in terms of the Hessian matrix itself, so that it is not tied to a particular function or primitive. A shape is considered isotropic if the eigenvalues of the Hessian matrix, evaluated at the shape centre, are of equal magnitude. The isotropy property is rotation invariant because the Hessian is rotation invariant [6], and scale invariant because scaling the function simply scales the Hessian (the relative magnitude of eigenvalues is preserved). The Hessian-based definition of isotropy is also compatible with the Gaussian-based definition of isotropy, since evaluating the Hessian at the centre of a Gaussian produces a matrix differing only in scale to $\mathbf{\Sigma}$ (see previous section).

Using inhomogeneous coordinates, $\mathbf{x} = \left[\begin{array}{cc} x & y \end{array}\right]^\top$, define function $i\left(\mathbf{x}\right)$ as being isotropic around $\mathbf{x} = \mathbf{0}$. It may be assumed that an arbitrary shape, $f\left(\mathbf{x}\right)$, is related to an isotropic shape $i\left(\mathbf{x}\right)$ by an affine transformation,

$$f\left(\mathbf{x}\right) = i\left(\mathbf{U}\mathbf{x}\right),$$

where $\mathbf{U}$ is of the form,

$$\mathbf{U} = k\mathbf{R}\left(\theta\right)\mathbf{A}.$$

The coordinate system is chosen so that the centre of $f\left(\mathbf{x}\right)$ is at the coordinate origin. The Hessian of shape $f\left(\mathbf{x}\right)$ is,

$$\frac{\partial^2 f\left(\mathbf{x}\right)}{\partial \mathbf{x}^2} = \frac{\partial^2 i\left(\mathbf{U}\mathbf{x}\right)}{\partial \mathbf{x}^2} = \mathbf{U}^\top \left[\begin{array}{cc} i_{xx}\left(\mathbf{U}\mathbf{x}\right) & i_{yx}\left(\mathbf{U}\mathbf{x}\right) \\ i_{xy}\left(\mathbf{U}\mathbf{x}\right) & i_{yy}\left(\mathbf{U}\mathbf{x}\right) \end{array}\right] \mathbf{U}.$$

Evaluating this equation at the shape centre point, $\mathbf{0}$, gives,

$$\left.\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2}\right|_{\mathbf{x}=\mathbf{0}} = \mathbf{U}^\top \begin{bmatrix} i_{xx}(\mathbf{0}) & i_{yx}(\mathbf{0}) \\ i_{xy}(\mathbf{0}) & i_{yy}(\mathbf{0}) \end{bmatrix} \mathbf{U}.$$

Since $i(\mathbf{x})$ is defined to be isotropic around $\mathbf{x} = 0$, the derivative matrix in the above equation is a scalar matrix of unknown scalar s, simplifying the equation to,

$$\begin{aligned} \left.\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2}\right|_{\mathbf{x}=\mathbf{0}} &= s\mathbf{U}^\top\mathbf{U} \\ &= sk^2\mathbf{A}\mathbf{R}(-\theta)\mathbf{R}(\theta)\mathbf{A} \\ &= sk^2\mathbf{A}^2. \end{aligned} \tag{6.2}$$

The Hessian therefore measures $sk^2\mathbf{A}^2$. Normalising the determinant and taking the square root gives $\mathbf{A}$. The angle of rotation, θ, is not measured and the scale, k, of the original $\mathbf{U}$ cannot be recovered from the Hessian due to the presence of unknown scale factor s (an unknown parameter of $i(\mathbf{x})$). Applying the inverse of $\mathbf{A}$ to the shape gives,

$$\begin{aligned} f\left(\mathbf{A}^{-1}\mathbf{x}\right) &= i\left(\mathbf{U}\mathbf{A}^{-1}\mathbf{x}\right) \\ &= i\left(k\mathbf{R}(\theta)\mathbf{A}\mathbf{A}^{-1}\mathbf{x}\right) \\ &= i\left(k\mathbf{R}(\theta)\mathbf{x}\right), \end{aligned}$$

which is an isotropic shape, since $i(\mathbf{x})$ is isotropic and isotropy is scale and rotation invariant.

This method effectively fits a second order surface to $f(\mathbf{x})$ in the vicinity of $\mathbf{x} = \mathbf{0}$ (and ignores the lower order components). Real images rarely contain purely second order shapes and contain noise. Fitting a second order surface to such an image by evaluating the Hessian at a single point is of no use. Computing a Gaussian scale space of the image removes pixel noise and smoothes local features. At the characteristic scale of a feature, a second order approximation is a reasonable

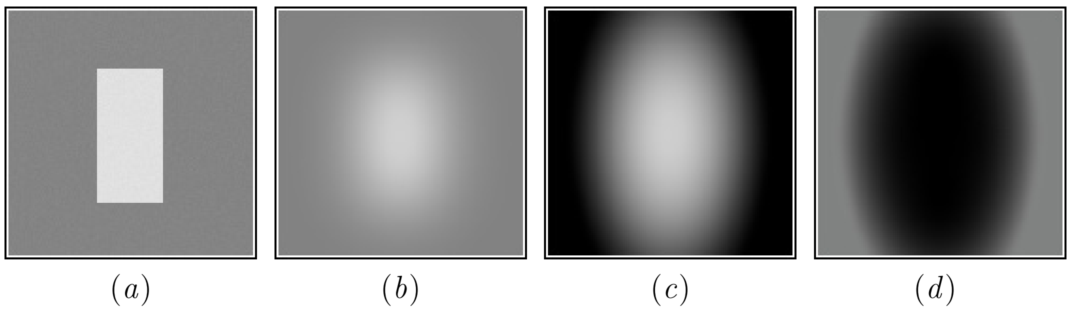


Figure 6.1: (a) A simple shape. (b) Characteristic scale image of (a). (c) A second order approximation to (b). (d) The difference between (b) and (c) (black indicates small difference).

approximation of the local feature shape, as illustrated in Figure 6.1. It can be seen that the second order function is a reasonable approximation in the feature area, but quickly becomes inaccurate outside the feature.

An isotropic function in Gaussian scale space may be defined as,

$$i_g(\mathbf{x}) = g(\mathbf{x}, \sigma) * i(\mathbf{x}).$$

In this definition, the isotropy property is scale dependant. A function that is isotropic at a particular scale may not be isotropic at another scale, because the Gaussian convolution removes different levels of detail from the shape at different scales.

It is not feasible to model a distorted version of an isotropic function in scale space as,

$$f(\mathbf{x}) = i_g(\mathbf{Ux}) = g(\mathbf{Ux}, \sigma) * i(\mathbf{Ux}),$$

because the distorted Gaussian, $g(\mathbf{Ux}, \sigma)$, cannot be produced without knowing $\mathbf{U}$. Instead, a distorted function in isotropic scale space may be observed as,

$$f(\mathbf{x}) = g(\mathbf{x}, \sigma) * i(\mathbf{Ux}).$$

The Hessian of this function is,

$$\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2} = \frac{\partial^2 g(\mathbf{x}, \sigma)}{\partial \mathbf{x}^2} * i(\mathbf{Ux}) = g(\mathbf{x}, \sigma) * \frac{\partial^2 i(\mathbf{Ux})}{\partial \mathbf{x}^2}.$$

This may be interpreted either as the Hessian of a smoothed version of the distorted function, or as a weighted average of the Hessian of the distorted function over an area proportional to the feature scale. The scale space thereby provides a method for fitting a second order surface to an arbitrary image structure in a stable manner. Evaluating the above function at $\mathbf{x} = \mathbf{0}$ does not, however, give a simple measure of the distortion as in Equation 6.2. Smoothing with an isotropic Gaussian affects the shape of a non-isotropic structure, since it does not average the shape information over an area of the same shape as the structure of interest. The result is that measuring the Hessian of an arbitrary shape in scale space provides only an approximate measure of the shape. The problem therefore lends itself to an iterative solution – if the distortion can be corrected using an approximate shape measure, then a second attempt at measuring the shape will yield better results because the shape of the Gaussian smoothing function will be more similar to the corrected structure shape.

The distortion of an isotropic function, $i(\mathbf{x})$, distorted by affine transformation $\mathbf{U}$, is iteratively estimated by means of the following procedure at each iteration,

$$\begin{aligned} f_j(\mathbf{x}) &= g(\mathbf{x}, \sigma) * i\left(\mathbf{U}\hat{\mathbf{U}}_j\mathbf{x}\right), && (6.3)\\ \hat{\mathbf{U}}_{j+1} &= \mathcal{H}_j^{-\frac{1}{4}}\hat{\mathbf{U}}_j\mathcal{H}_j^{-\frac{1}{4}}, && (6.4)\end{aligned}$$

where f_j is the measured function at iteration j, $\mathcal{H}_j$ is the Hessian of $f_j(\mathbf{x})$, evaluated at $\mathbf{x} = \mathbf{0}$, and $\hat{\mathbf{U}}_j$ is the affine correction transformation (a symmetric matrix with determinant of 1) and represents the inverse of the distortion estimate at iteration j (with $\hat{\mathbf{U}}_0 = \mathbf{I}$). Eventually this process will converge such that f_j is isotropic and $\mathcal{H}_j$ is scalar. If f_j is isotropic, then $i\left(\mathbf{U}\hat{\mathbf{U}}_j\mathbf{x}\right)$ is isotropic. In this case, convolution with the isotropic Gaussian function has no affect on the isotropy of $i\left(\mathbf{U}\hat{\mathbf{U}}_j\mathbf{x}\right)$ (other than to adjust scale), because both functions are isotropic. The matrix, $\hat{\mathbf{U}}_j$, is then equal to the inverse of the $\mathbf{A}$ component of $\mathbf{U}$. A correction transformation has therefore been found that is covariant with $\mathbf{U}$, as long as the scale, σ, is selected in a covariant manner.

Note the method used to update the correction transformation in Equation 6.4. This method prevents an unwanted rotation component from being introduced into $\hat{\mathbf{U}}$.

6.3 Practical Issues

The Hessian matrix is predicted to be less stable than the second moment matrix due to the absence of an averaging window applied after differentiation. For this reason it is recommended that a small averaging filter be applied to improve stability. Implementations presented in this thesis make use of a Gaussian filter with scale parameter $\sigma = 1$ and truncated to a 3×3 pixel window to average the Hessian matrix over a small area. This has a practically insignificant effect on the scale at which the Hessian is measured, but results in more stable and noise tolerant measurements.

The Hessian matrix is not positive definite. In a saddle point the Hessian matrix will have one positive and one negative eigenvalue, in which case the Hessian can not be properly scalar. The definition of isotropy does not require the sign of gradients to be the same, only their magnitudes, and so isotropy may still be achieved. Simply negating the negative eigenvalue of the measured Hessian matrix is sufficient to deal with the problem. One possible way to do this is to compute the singular value decomposition (SVD) of the Hessian matrix, thereby accessing the eigenvalues directly. The SVD of a 2×2 symmetric matrix is trivial to compute. It can also be used to compute the square root of the matrix and the isotropy measure (see Chapter 4, Section 4.3.2), which form part of the adaptation algorithm (Section 4.4).

6.4 Comparison of the Complexity of the Hessian and Second Moment Operators

Both the Hessian matrix and the Second Moment matrix can be computed from an image by applying a series of filters to the image. The scale operator and windowed integration operation can be implemented as Gaussian filters, which are most efficiently realised as IIR filters [14, 59]. The differentiation operations can be implemented as 3×3 kernel filters. The complexity of all these filters is linear in the number of image pixels processed. The only difference between computing the Hessian matrix and the Second Moment matrix is the arrangement of the above filters.

To compute these operators at a single point in an image (as is done repeatedly during affine adaptation), the filters need only be applied to the image neighbourhood that has a significant influence on the point of interest – this is referred to as the filter support region. The first filter must be applied to the sum of the support regions of all the filters required to compute an operator. Once one filter has been applied, its support region can be discarded, leaving a smaller area to process for subsequent filters. The derivative kernels require a 3 pixels wide support area. IIR Gaussian filters technically require infinite support (because the Gaussian is non-zero for all real coordinates), though the region within a radius of 3σ accounts for over 99% of the filter response. The total support width for a Gaussian filter is therefore specified as 6σ.

The number of operations per pixel required by each filter is listed in Table 6.1. Each derivative kernel requires a different number of operations per pixel, due to the presence of zero elements in the kernels. The derivative kernels are given in Figure 6.2. The constant k_g depends on the specific Gaussian filter implementation.

Filter	Label	Op's per pixel
$\partial I/\partial x$	c_x	6
$\partial I/\partial y$	c_y	6
$\partial^2 I/\partial x^2$	c_{xx}	9
$\partial^2 I/\partial y^2$	c_{yy}	9
$\partial^2 I/\partial x \partial y$	c_{xy}	4
$g(\mathbf{x}, \sigma)$	$c_{g\sigma}$	k_g
$g(\mathbf{x}, 1)$ truncated to 3×3	c_{g1}	9

Table 6.1: Number of operations per pixel for filters.

-1	0	1
-1	0	1
-1	0	1

c_x

1	1	1
0	0	0
-1	-1	-1

c_y

1	-2	1
1	-2	1
1	-2	1

c_{xx}

1	1	1
-2	-2	-2
1	1	1

c_{yy}

-1	0	1
0	0	0
1	0	-1

c_{xy}

Figure 6.2: Image derivative kernels.

The Hessian operator consists of the following processing steps:

1. Filter to the feature scale, σ, with a Gaussian filter, at a cost of $c_{g\sigma}$ operations per support pixel. A support region of 6σ pixels is required.

2. Compute second order partial derivatives (three filters) at a cost of c_{xx}, c_{yy} and c_{xy} operations per support pixel. A support region of 3 pixels is required.

3. Apply 3×3 averaging filter to each of the three derivative images, at a cost of c_{g1} operations for each of the three derivative images. A support region of 3 pixels is required.

The number of operations required to compute the Hessian matrix at a pixel is,

$$\begin{aligned} c_H &= c_{g\sigma}(6\sigma + 3 + 3) + (c_{xx} + c_{yy} + c_{xy})(3 + 3) + 3c_{g1} \\ &= k_g(6\sigma + 6) + 159. \end{aligned}$$

The Second Moment operator consists of the following processing steps:

1. Filter to differentiation scale σ_D with a Gaussian filter, at a cost of $c_{g\sigma}$ operations per support pixel. A support region of $6\sigma_D$ pixels is required.

2. Compute the first order partial derivative images L_x and L_y, at a cost of c_x and c_y operations per support pixel. A support region of 3 pixels is required.

3. Multiplying the derivative images to produce images L_x^2, L_y^2 and L_xL_y, at a cost of one operation per pixel for each product image.

4. Filter each derivative product image with a Gaussian with scale σ_I to compute windowed integration, at a cost of $c_{g\sigma}$ per pixel per image. A support region of $6\sigma_I$ pixels is required.

The number of operations required to compute the Second Moment matrix at one pixel is,

$$\begin{aligned} c_S &= c_{g\sigma}\left(6\sigma_I + 6\sigma_D + 3\right) + \left(c_x + c_y\right)\left(6\sigma_I + 3\right) + 3\left(6\sigma_I\right) + 3c_{g\sigma}\left(6\sigma_I\right) \\ &= k_g\left(24\sigma_I + 6\sigma_D + 3\right) + 90\sigma_I + 36. \end{aligned}$$

Given that for the same feature both σ_I in the Second Moment operator and and σ in the Hessian operator are set to the characteristic scale of the feature, it is clear that the Hessian matrix requires significantly less computational effort than the Second Moment matrix.

6.5 Experimental Evaluation

In this section the Hessian matrix and second moment matrix are compared in terms of their effectiveness and efficiency in adapting local image features.

Label	Saliency function (E)	Scale function (S)	Shape estimator (A)
He_H	Det of Hessian	Det of Hessian	Hessian
He_S	Det of Hessian	Det of Hessian	SMM
Ha_H	Harris	Laplacian	Hessian
Ha_S	Harris	Laplacian	SMM

Table 6.2: Test Feature Extractor Configurations.

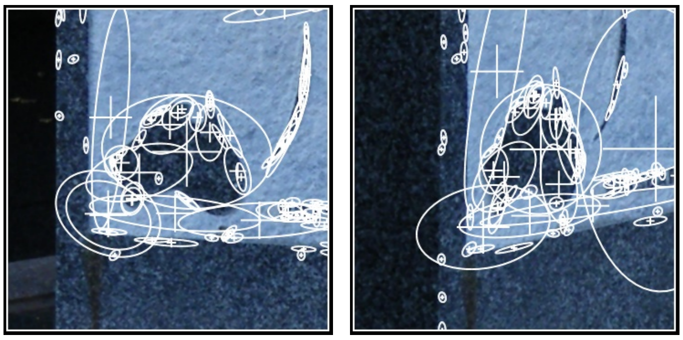
6.5.1 Evaluation Method

Four feature extractors were constructed using the algorithm presented in Chapter 4, Section 4.4, each using a different combination of estimation functions. These extractors are listed in Table 6.2. All extractors used the orientation selection method and SIFT descriptor described in [38]. Figure 6.3 shows the output of the four extractors on corresponding sections of a pair of images. Note that the Hessian and second moment matrices produce different shapes for the same feature, but that the shape is none the less covariant between images.

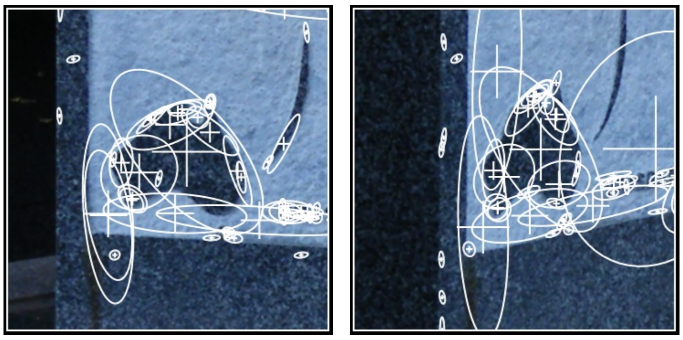
Two evaluation methods are used to compare the above extractors. The first is the repeatability test procedure and data described in [45] and Chapter 3, Section 3.3.1. The second is the epipolar geometry computation task described in Chapter 7, Section 7.2 (this evaluation method makes use of the WiDense algorithm, developed in Chapter 7). Only the two datasets with the simplest geometry (no. 5 and 6) were used for this test, because the other datasets are so challenging that they do not give useful results for the set of extractors tested here. Ground truth was generated using the MSER extractor.

6.5.2 Results

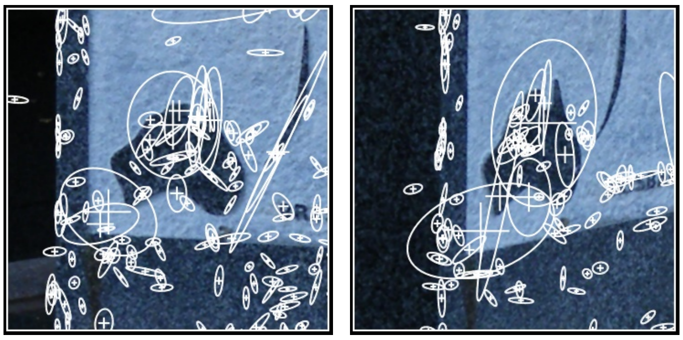
The repeatability results were analysed by means of paired T-tests. A paired T-test comparing the Hessian extractors (He_H and He_S) is listed in Table 6.3,



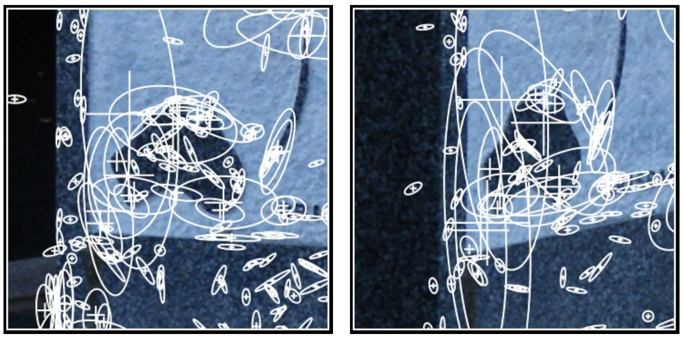
He_H



He_S



Ha_H



Ha_S

Figure 6.3: Image sections showing the output of the evaluation extractors.

and a paired T-test comparing the Harris extractors (Ha_H and Ha_S) is listed in Table 6.4.

	mean(He_H)	mean(He_S)	mean(He_H / He_S)	p
repeatability (%)	58.67	57.99	1.01	0.0680
correspondences	1711.93	1564.13	1.12	0.0000
matching score (%)	20.90	21.94	0.95	0.0295
correct matches	492.67	467.80	1.05	0.0002
efficiency (n/s)	95.93	16.34	5.92	0.0000

Table 6.3: Paired T-test comparing results of He_H and He_S.

	mean(Ha_H)	mean(Ha_S)	mean(Ha_H / Ha_S)	p
repeatability (%)	38.30	47.53	0.79	0.0000
correspondences	528.83	937.13	0.53	0.0000
matching score (%)	14.98	17.33	0.87	0.0001
correct matches	178.03	293.30	0.58	0.0000
efficiency (n/s)	27.10	12.93	2.08	0.0000

Table 6.4: Paired T-test comparing results of Ha_H and Ha_S.

Table 6.5 lists the results for the epipolar geometry computation task with the error threshold set to $t = 16$. The table lists the average number of successful epipolar geometry computation trials for each extractor in each dataset, as well as the average success rate over all datasets. Figure 6.4 shows the average success rate over all datasets at each of the tree error thresholds $t = 4$, $t = 16$ and $t = 64$.

6.5.3 Discussion

The discussion of results focuses on comparing the performance of extractors using different affine shape measures. The determinant of Hessian extractors, He_H and He_S, are compared and the Harris extractors, Ha_H and Ha_S, are compared separately. The Hessian and Harris extractors have been compared to each other in previous studies.

A paired T-test comparing the Hessian extractors (He_H and He_S) is summarised

set	n_f	$\mathrm{Ha_S}$	$\mathrm{Ha_H}$	$\mathrm{He_S}$	$\mathrm{He_H}$
5	46	16.14	5.26	10.09	14.61
6	41	12.52	4.38	12.1	18.76
total	87	28.66	9.64	22.19	33.37
%		32.94	11.08	25.51	38.36

Table 6.5: Results for the epipolar geometry computation task. The average number of successful epipolar geometry computation trials with error threshold $t = 16$ are listed.

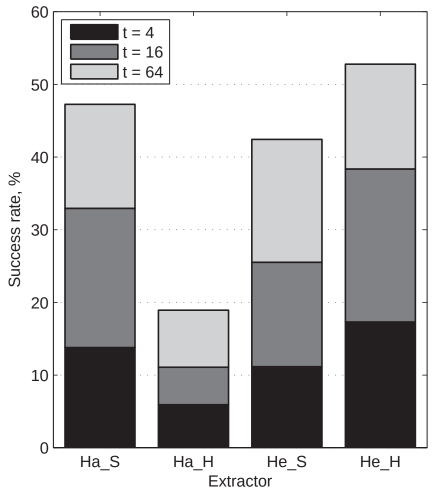


Figure 6.4: Average success rates for the epipolar geometry computation task.

in Table 6.3. In terms of repeatability, matching score and the number of correct matches, there is no significant difference between these two extractors on average. In terms of efficiency, the extractor using the Hessian matrix to estimate affine shape ($\mathrm{He_H}$) consistently outperforms the extractor using the second moment matrix ($\mathrm{He_S}$) by a factor as high as 6.6 times and 5.9 times on average. Using the Hessian matrix resulted in a small increase of approximately 12% in the number of correspondences. The difference in performance between the two extractors is consistent across all types of tests. In the epipolar geometry estimation task, correspondences generated using $\mathrm{He_H}$ are successfully used to compute the epipolar geometry in approximately 1.5 times as many cases as correspondences generated using $\mathrm{He_S}$. This indicates that the Hessian-based shape adaptation is more likely to produce useful results in practical problems than the second moment matrix-based method.

A paired T-test comparing the Harris extractors (Ha_H and Ha_S) is summarised in Table 6.4. These two extractors produced significantly different results. The extractor using the Hessian matrix to estimate affine shape (Ha_H) consistently achieved lower repeatability scores (79%), matching scores (87%) and produced approximately half the number of correspondences and correct matches. Despite lagging in the other metrics, the Ha_H extractor is on average twice as efficient at producing correct matches as the Ha_S extractor. In the epipolar geometry estimation task, using Ha_H produces useful results in only a third as many cases as Ha_S.

This difference in behaviour of Hessian matrix-based affine adaptation between the Harris and Hessian extractors may be attributed to the fact that the Harris extractor selects corner features more often than the Hessian extractor, while the Hessian extractor predominantly selects blob-like structures. The Hessian matrix should be computed at the centre of a blob, as outlined in Section 6.1, and may be unstable or inaccurate otherwise. A second order approximation without the first order terms is also not sufficiently accurate at a corner. The results indicate that the Hessian matrix is very effective and efficient in affine adaptation of blob-like features, but is much less effective for corner features.

6.6 Chapter Summary

In this chapter it is shown that the Hessian matrix can be used to estimate the affine shape of local image features using an iterative approach, similar to how the second moment matrix is commonly used. A blob extractor that makes use of the Hessian matrix requires on average 5.9 times less processing time than an extractor using the second moment matrix, while exhibiting the same performance in terms of repeatability, matching score and the number of correspondences and correct matches. This reduction in computation time is primarily attributed to the fact that fewer filter stages are required to compute the Hessian matrix, compared to the second moment matrix. The Hessian matrix-based method also leads to useful epipolar geometry estimates in 1.5 times the number of cases, compared to the second moment matrix-based method.

When combined with a corner extractor, the Hessian matrix produced lower repeatability (79%), fewer correspondences (53%), fewer correct matches (58%) and fewer useful epipolar geometry estimates (34%) than the second moment matrix. This is due to the fact that the Hessian matrix is less stable and accurate when computed in regions other than at the centre of a blob-like image region. Despite the reduction in correspondence counts, using the Hessian matrix still produced correspondences approximately twice as efficiently as the second moment matrix.

Chapter 7

Dense Optical Flow across Uncalibrated Wide Baseline Views

The affine covariant feature extractors investigated and developed in earlier chapters are robust and effective in extracting correspondences between wide baseline views. Despite the quality of modern correspondence extraction methods, they can not provide sufficient correspondences in all camera and scene configurations. If cameras are too sparsely placed, produce low resolution images, or if the scene contains few distinguishable features, then wide baseline matching techniques may not be able to produce sufficient correspondences to allow computation of the camera geometry.

This chapter presents the Uncalibrated Wide Baseline Dense Optical Flow algorithm, called WiDense. It is a method for extracting large numbers of accurate correspondences between a pair of views using only a set of putative wide baseline correspondences as input. The new method does not rely on any knowledge of the camera geometry, only the information contained in matches produced through wide baseline matching. It can produce useful output even when the input set of correspondences is not sufficient to compute the epipolar geometry. The WiDense algorithm is presented in Section 7.1.

In Section 7.2 a method for evaluating wide baseline correspondence extraction systems is presented. The evaluation attempts to compute the epipolar geometry of a set of scenes using the output of a given wide baseline matching system. The results are presented in terms of average success rates. This evaluation method is used in Section 7.3 to compare the performance of the WiDense algorithm to existing wide baseline matching techniques. It is shown through this evaluation that the correspondences generated by the WiDense algorithm can be used to compute the epipolar geometry in many cases where existing wide baseline methods alone do not provide sufficient reliable correspondences.

7.1 The WiDense Algorithm

Matching features across widely separated views usually involves comparing feature sets containing a large proportion of features that are not common to both views. Each feature must therefore be described with a large amount of information in order to distinguish it from the interfering features. This means that each feature that is successfully matched contains simpler, smaller scale features within its support region. These sub-features do not contain enough information to be matched unambiguously across views, but can be exploited once the primary feature has been matched.

A matched pair of affine covariant features provides much more information than simply a corresponding pair of points. They also contain normalisation transformations that approximately map the originating local image regions to a common coordinate frame. These affine normalisation transformations can be combined into a transformation that maps one image to the other. This transformation is valid over the whole plane on which the feature is located. A matched pair of affine covariant features therefore gives an affine planar transformation between the two images of the plane containing the feature.

The Uncalibrated Wide Baseline Optical Flow, or WiDense, algorithm presented in this section takes advantage of the affine mapping provided by matched features and the information contained in their local image regions. The known local planar transformation is first refined and then used to constrain the search space, allowing the search for fine scale feature correspondences. The surrounding neighbourhood of the matches is also explored using this transformation, in order to find more correspondences.

The term, "dense correspondences", is used slightly differently in different contexts. In the context of short baseline stereo, for example, it refers to correspondences for every pixel in an image. Another example is to subdivide an image in a regular grid and find a correspondence for every cell in the grid. In this thesis, dense correspondence means that correspondence has been found for every image region that can be aligned. Image regions that contain no information (homogeneous) provide no means of finding correspondences. Furthermore, in a wide baseline scenario, occlusions may be very large and the commonly visible part in the scene may account for a small proportion of image area. It is therefore not possible to match every part of an image in these scenarios.

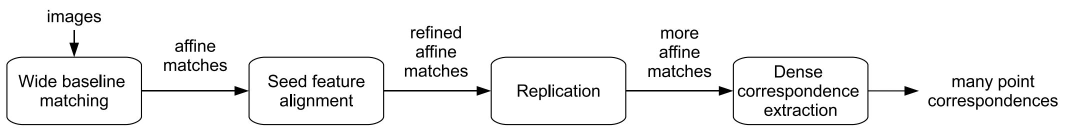


Figure 7.1: The uncalibrated wide baseline dense correspondence extraction process.

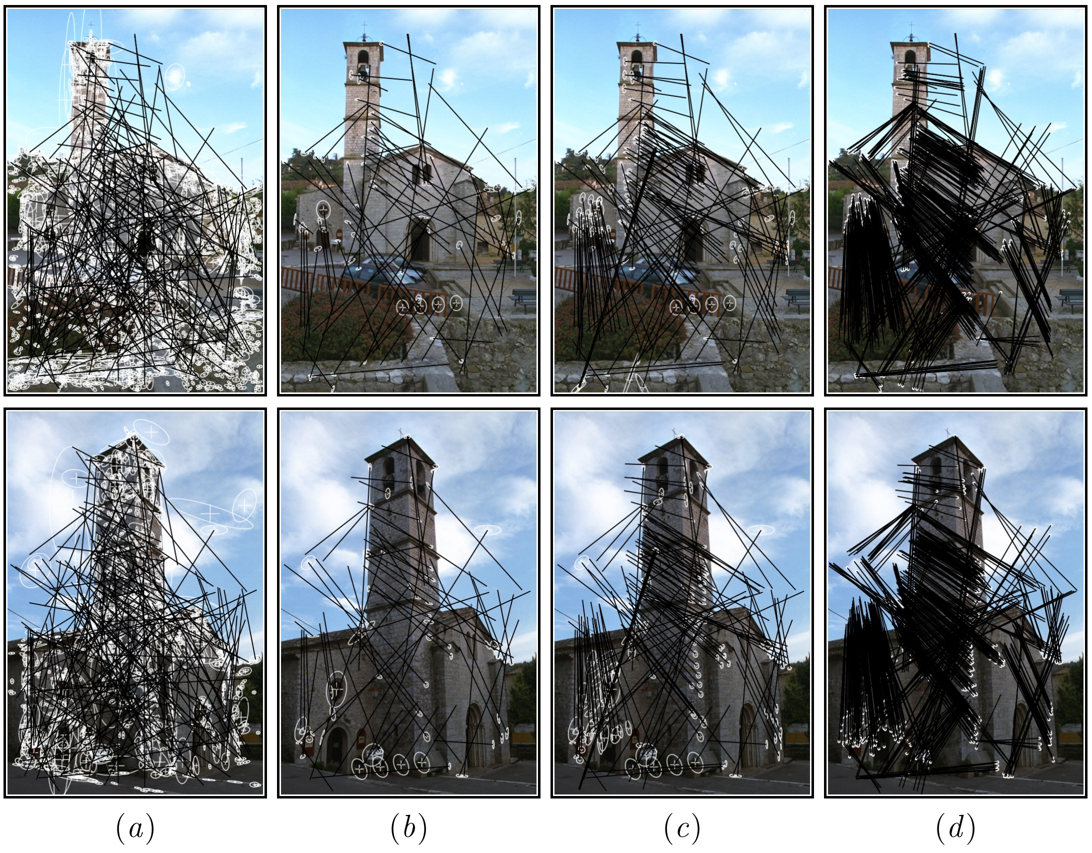


Figure 7.2: Example output from the different stages of the WiDense algorithm. Features are indicated by white ellipses and correspondences are indicated by black lines joining the feature positions in each image. (*a*) Phase one: Hessian Affine features and putative correspondences. (*b*) Phase two: Aligned seed correspondences. (*c*) Phase three: Replicated correspondences. (*d*) Phase four: Dense correspondences.

7.1.1 Overview

The WiDense algorithm consists of four phases. Figure 7.1 presents a diagram of the system and Figure[1] 7.2 shows the output of each phase on an example image pair.

The first phase is the conventional wide baseline matching step. It consists of extracting and matching affine covariant features using any affine covariant feature extractor, descriptor and matching routine. This phase generates a set of putative correspondences that may contain a large proportion of incorrect matches, depending on the scene complexity. In the example in Figure 7.2(a), the set of 210 putative correspondences contains approximately 9% correct matches.

The second phase consists of properly aligning the image regions associated with each matched pair of features. This phase is similar to the match refinement stage in [18], however the alignment method used is significantly different. The output of this step is a set of seed features with refined normalisation transformations. Some incorrect matches are eliminated by removing correspondences that cannot be aligned with sufficiently low error. This phase is referred to as seed correspondence alignment, and is discussed in Section 7.1.2. In the example in Figure 7.2(b), the set of putative correspondences has been reduced to 67 correspondences, containing approximately 28% correct matches.

The third phase involves searching the neighbourhood of each seed correspondence for other regions that are related by approximately the same transformations as the seed features. The alignment of each new correspondence is refined to accommodate limited changes in surface orientation. This process is referred to as correspondence replication and is presented in Section 7.1.3. In the example in

[1]Images sourced from `http://www.robots.ox.ac.uk/~vgg/data/valbonne/images.tar.gz`.

Figure 7.2(c), the set of correspondences has been expanded to 155 correspondences, of which approximately 45% are correct.

The final step is to select small scale features from each corresponding pair of affine features and to align them precisely. This is presented in Section 7.1.4. In the example in Figure 7.2(d), a total of 616 point correspondences were extracted, of which 60% are correct.

The algorithm produces a large number of highly accurate point correspondences, containing a relatively small proportion of outliers. Further example output and basic observations regarding the algorithm are presented in Section 7.1.5. Lengthy derivations are presented separately in Appendix A to simplify the algorithm presentation.

7.1.2 Seed Region Alignment

Affine covariant features provide a normalisation transformation that maps a pair of corresponding features to a common coordinate frame. These transformations are computed independently for each feature from images taken under very different conditions. As a result, they are not highly accurate. This section discusses a procedure for accurately aligning corresponding image regions, given the approximate normalisation transformation for each region. It will be assumed that an affine transformation is a sufficient local approximate model for relating two images of the same surface. As a by-product, the seed alignment process removes some incorrect matches when they cannot be aligned with sufficiently low error.

Expected Alignment Error

The initial normalisation transformations included in a matched pair of features can be expected to contain significantly large error. The nature and severity of this error is discussed here.

If the image region is symmetric to a significant degree, then it is likely that the selected orientations of the two regions do not correspond. The robustness of the SIFT descriptor makes it possible to match features despite such an error. It is therefore possible that there is gross error in terms of region rotation. An example case that suffers this problem is presented in Figure 7.3. If a rotation invariant descriptor was used for matching, then the feature orientation may not be available at all.

The anisotropic scaling components, $\mathbf{A}\left(q,\phi\right)$, of the affine transformations are unlikely to be grossly inaccurate, but can be expected to contain limited error, even in ideal conditions. Larger regions may suffer from projective transformation that cannot be fully accounted for by anisotropic scaling. This effect is usually negligible, however. Image regions that contain more than one plane cannot be aligned perfectly using only a linear transformation, and hence will suffer from varying degrees of error on each plane.

The translation component of the affine transformations (or the feature location) can be expected to be reasonably accurate. The most significant source of translation error is non-planar regions of the image.

The image intensity can vary to a great extent between views due to different camera exposure settings and lighting conditions. Feature extractors and descriptors are generally very robust to intensity variations (except when saturation occurs). It is therefore likely that correspondences may be found with a large difference in intensity. Although only geometric alignment is of interest, image intensities must also be aligned in order for the geometric alignment process to obtain useful error estimates. Due to the local extent of the image region, a linear mapping of intensity is sufficient to compensate for photometric differences.

Algorithm

The seed region alignment algorithm has the following steps:

1. Select which feature is used as the template and which feature is to be registered to the template.
2. Compute normalised template image patch.
3. Initialise the registration transformation.
4. Coarse orientation selection.
5. Iterative inverse compositional alignment.
6. Check that results are reasonable and discard matches that are not within error and geometric bounds.
7. Subdivide regions with excessively eccentric shape.

Each step is now discussed in detail. Figure 7.3 illustrates the process up to step 5.

1) Template Selection: The image alignment process operates by using one image as a static template image and the second image as a registration image that is to be mapped onto the template image. The choice of which feature to use to generate the template is made based on the eccentricity, q, of the normalisation

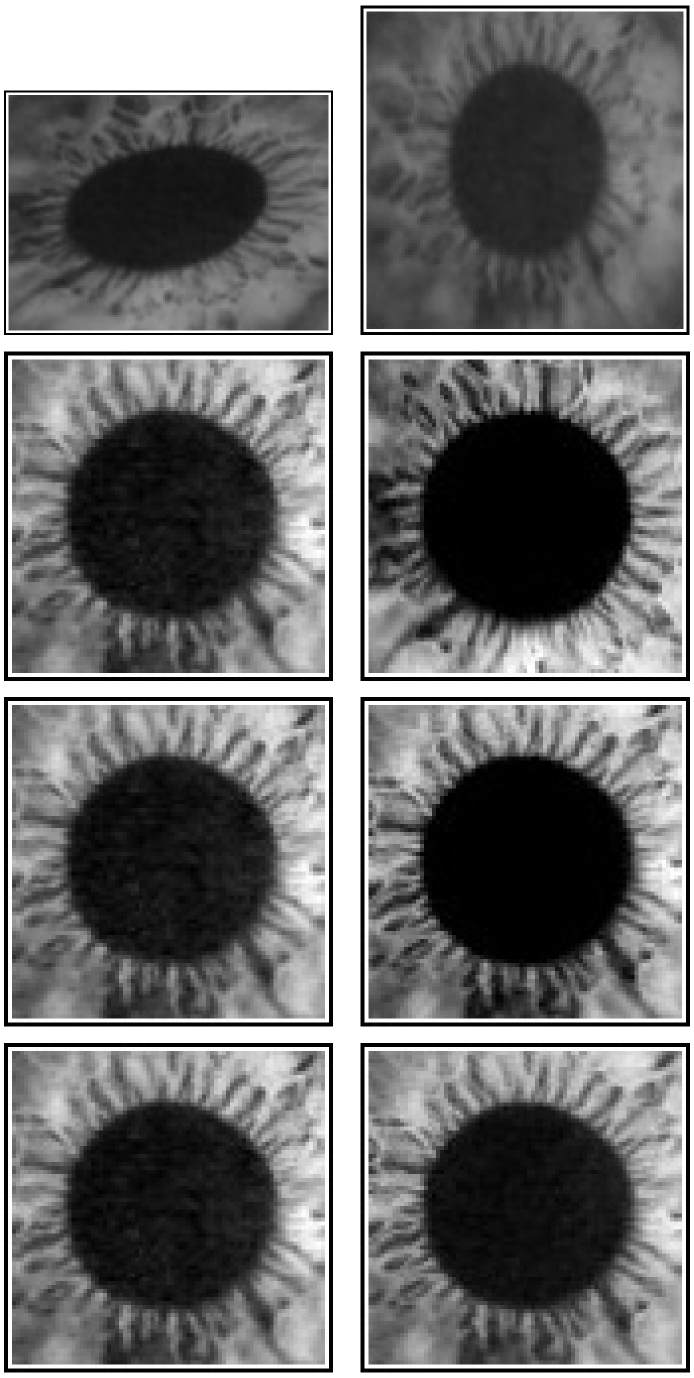


Figure 7.3: The seed alignment algorithm. The left column is chosen as the template feature, and the right column is the corresponding registration feature (step one). Row 1 – The original image regions, I_t and I_r. Row 2 – Normalised template $I_{tn}(\mathbf{x})$ and initial $I_r(\mathbf{H}_r\mathbf{x})$ (step two and three). Row 3 – $I_r(\mathbf{H}_r\mathbf{x})$ after computing rough orientation estimate (step four). Row 4 – $I_r(\mathbf{H}_r\mathbf{x})$ after inverse compositional alignment (step five).

transformations. The feature with eccentricity furthest from one (most eccentric) is used as template, since the normalised image will be most affected by anisotropic sampling distortion. This distortion effect is counteracted to some extent in the process of applying an anti-aliasing filter in step 2. In the following, the chosen template feature is referred to as $\mathbf{H}_t$ and its source image as I_t. The registration feature and its image are referred to as $\mathbf{H}_r$ and I_r.

2) Compute Normalised Template Image: The objective of this step is to compute a normalised template image with dimensions $2r_t + 3 \times 2r_t + 3$, to be used in the image alignment process. The author's implementation of this algorithm uses $r_t = 20$, which results in gradient images with a width of 41 pixels in step 5. These values provide a sufficient number of samples for the alignment process, while keeping the processing costs low. The normalisation process consists of anisotropic scaling of the feature support region, applying an anti-aliasing filter, down-sampling to the desired image size and photometric normalisation.

The following transformation is applied to the feature support region to remove anisotropic scaling and shift the feature to the origin,

$$\mathbf{H}_n = \mathbf{T}\left(t_{xt}, t_{yt}\right)\mathbf{A}\left(q_t, \phi_t\right)\mathbf{K}\left(q_t\right).$$

Scaling by $\mathbf{K}\left(q_t\right)$ ensures that the largest eigenvalue of $\mathbf{A}\left(q_t, \phi_t\right)\mathbf{K}\left(q_t\right)$ is 1, thereby preventing aliasing during image transformation. Note that no rotation component has been included in $\mathbf{H}_n$. This is because the orientations supplied by the features are assumed to be unreliable or unavailable (see Section 7.1.2). The orientation is recovered in step 4.

The affine rectified image patch is then computed as, $I_a\left(\mathbf{x}\right) = I_t\left(\mathbf{H}_n\mathbf{x}\right)$, over the coordinate range $x, y \in \left[-k_t q_t^{-1} - 1, k_t q_t^{-1} + 1\right]$ (the local extent of the feature). At this point the rectified image needs to be scaled by a factor of $s_n = r_t / k_t q_t$. A Gaussian anti-aliasing filter with parameter $\sigma = s_n/3$ is first applied to the rectified image. The scaled, rectified image is then computed as, $I_s(\mathbf{x}) = I_a\left(\mathbf{K}\left(s_n^{-1}\right)\mathbf{x}\right)$, over the coordinate range $x, y \in [-r_t - 1, r_t + 1]$. In cases where the feature is smaller than the desired patch size, the rectified image is not scaled ($I_s(\mathbf{x}) = I_a(\mathbf{x})$).

The final step in preparing the template patch is photometric normalisation. The parameters of a linear intensity mapping are computed as,

$$\begin{aligned} l &= \min\left(I_s\right), \\ h &= \max\left(I_s\right), \\ i_{ut} &= \tfrac{255}{h-l}, \\ i_{vt} &= -l i_{ut}, \end{aligned}$$

and the final normalised template image is computed as, $I_{tn} = i_{ut} I_s + i_{vt}$. This linear mapping maps the minimum intensity value to 0 and the maximum to 255. The parameters of this mapping are stored along with the template feature for use in later processing steps.

3) Initialise Registration Transformation: The registration feature transformation needs to be modified slightly in order to be compatible with the template image constructed in the previous step. The registration transformation is initialised as,

$$\mathbf{H}_r = \mathbf{T}\left(t_{xr}, t_{yr}\right)\mathbf{A}\left(q_r, \phi_r\right)\mathbf{K}\left(k_r r_t^{-1}\right).$$

If the template image is smaller than r_t, then the registration transformation is initialised as,

$$\mathbf{H}_r = \mathbf{T}\left(t_{xr}, t_{yr}\right)\mathbf{A}\left(q_r, \phi_r\right)\mathbf{K}\left(k_r k_t^{-1} q_t^{-1}\right).$$

The intensity mapping for the registration image is simply copied from the template image parameters, $i_{ur} = i_{ut}$, $i_{vr} = i_{vt}$. The intensity mapping is not computed using the same procedure as step 2 because simply selecting the minimum and maximum intensity is sensitive to spurious pixel values.

4) Coarse Orientation Selection: As mentioned in Section 7.1.2, the orientations of the features may be inconsistent. For this reason, the orientation is found by means of a coarse exhaustive search. The registration transformation is computed at twenty different rotation angles, evenly spaced over a full rotation, according to,

$$\mathbf{H}_{ri} = \mathbf{H}_r \mathbf{R}\left(\theta_i\right), \quad i \in \mathbb{N}\left[1, 20\right].$$

The mean squared error between the template image and the transformed registration image is measured at each angle. A parabola is fit to the error values around the lowest error value and used to interpolate the estimated angle of minimum error, $\hat{\theta}$. The registration transformation is then updated using $\hat{\theta}$,

$$\mathbf{H}_r = \mathbf{H}_r \mathbf{R}\left(\hat{\theta}\right).$$

5) Iterative Alignment: This step consists of applying the inverse compositional image alignment algorithm [3] to refine the registration transformation $\mathbf{H}_r$ in order to minimise the difference between $I_{tn}\left(\mathbf{x}\right)$ and $I_r\left(\mathbf{H}_r\mathbf{x}\right)$. The algorithm in [3] is extended to align image intensities using a linear mapping (parameters i_{ur} and i_{vr}) at the same time as geometric alignment is performed. The derivation of the combined geometric and photometric inverse compositional image alignment algorithm is presented in Section A.2. After alignment, the scale factor r_t is removed form the transformations.

6) Consistency Check: Inverse compositional alignment is not guaranteed to succeed, though it is likely to succeed in the majority of correct matches and even in some cases where the matches are not correct. In order to eliminate correspondence where alignment has failed, the final RMS error, q_r and k_r are checked to be within reasonable bounds. A significant proportion of incorrect matches may be eliminated by checking the mean squared error, at the risk of also eliminating correct matches. Through trials it has been determined that a RMS error threshold of 40 rejects the majority of incorrect matches while ensuring that few successfully aligned correct matches are rejected. In future research it may be possible to choose this threshold automatically based on the image contents, thereby allowing the algorithm to reject incorrect matches with greater reliability.

The q_r and k_r are checked to detect grossly deformed regions that coincidentally do not result in large RMS error. The thresholds for these parameters need only be set at the extremes of reasonable values. The threshold for q_r was chosen to be 0.05. The threshold for k_r was chosen to be $(d_{\min}/4)^2$, where $d_{\min}$ is the minimum image dimension.

It can be seen from Figure 7.2 and the examples in Section 7.1.5 that this step can eliminate a significant proportion of the incorrect putative matches.

7) Eccentric Feature Subdivision: The feature normalisation transformations of a corresponding pair of features are often more eccentric (smaller q parameter) than the transformation directly relating the two feature image regions. This is because the affine shape of each feature is computed from the distribution of image gradients, and not from the orientation of the surface relative to the viewpoint. The result is that the transformations applied during image alignment cause excessive amounts of distortion to the images, which affects alignment accuracy.

One possible solution is to find the direct transformation between the two regions, and to use this transformation to directly align images instead of using the normalisation approach. Because it is convenient to process rectangular image regions rather than any other shape, using this approach can result in neighbouring regions being included in the alignment process. These regions may not be coplanar with the region of interest and can adversely affect alignment accuracy.

The solution proposed here subdivides very eccentric features across their longest axis so that the resulting features are less eccentric and cover the same image

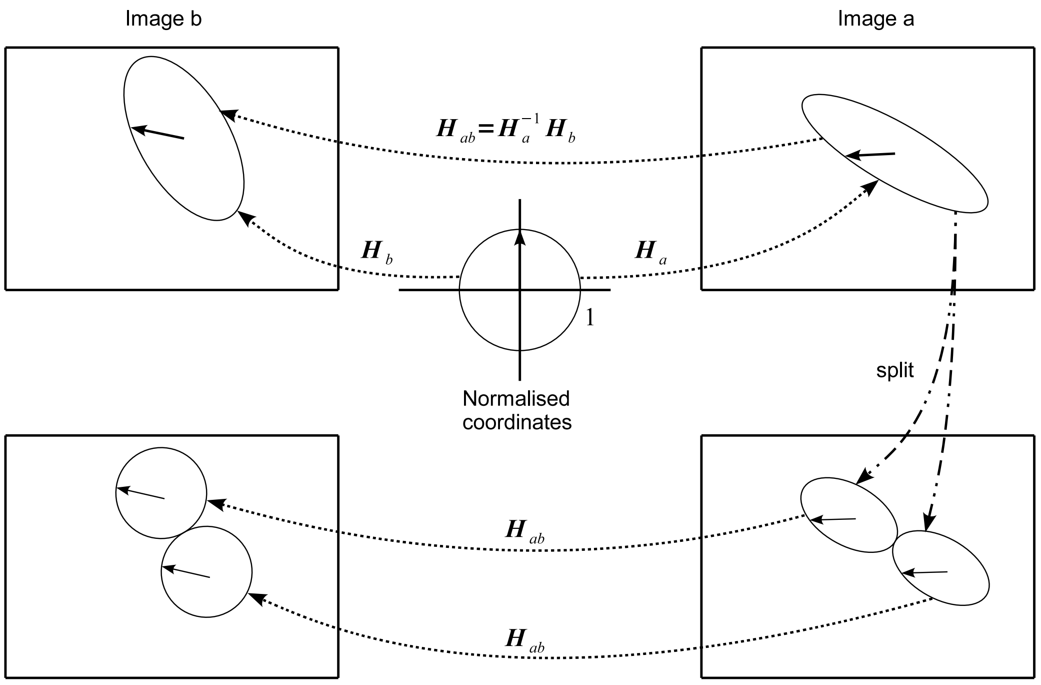


Figure 7.4: The eccentric feature subdivision process.

surface. The result is that each image is distorted to a limited degree, instead of one image suffering the full effect of the direct transformation, or both images being warped excessively.

Subdividing a pair of features is only useful if the following three criteria are met:

1. $k >= 24$ for both features.
2. $q_1 q_2 < 0.69$.
3. $q_{12} < q_d$, where q_{12} is the eccentricity parameter of matrix $\mathbf{H}_{12} = \mathbf{H}_1^{-1}\mathbf{H}_2$ and q_d is the eccentricity parameter of matrix $\mathbf{A}_d\left(q_d, \phi_d\right) = \mathbf{A}\left(q_1, \phi_1 + \theta_1\right)\mathbf{A}\left(q_2, \phi_2 + \theta_2\right)$.

The derivation of these criteria can be found in Section A.4.

Correspondences that meet the above criteria are subdivided by splitting the most eccentric feature across its longest axis and projecting the two resulting features to the other image. The feature subdivision procedure is illustrated in Figure 7.4 and discussed below. The derivation of the procedure is documented in Section A.5.

The most eccentric feature is labelled $\mathbf{H}_a$ and the corresponding feature is labelled $\mathbf{H}_b$. Feature $\mathbf{H}_a$ is split in to features $\mathbf{H}_{a1}$ and $\mathbf{H}_{a2}$ by computing the parameters of the new features as follows:

$$\begin{aligned}
t_{x\delta} &= -0.5 k_a q_a^{-1} \sin\left(\phi_a\right), \\
t_{y\delta} &= 0.5 k_a q_a^{-1} \cos\left(\phi_a\right), \\
t_{xa1} &= t_{xa} + t_{x\delta}, \\
t_{ya1} &= t_{ya} + t_{y\delta}, \\
t_{xa2} &= t_{xa} - t_{x\delta}, \\
t_{ya2} &= t_{ya} - t_{y\delta}, \\
k_{a1} &= k_{a2} = \frac{1}{\sqrt{2}} k_a, \\
q_{a1} &= q_{a2} = \sqrt{2} q_a, \\
\theta_{a1} &= \theta_{a2} = \theta_a, \\
\phi_{u1} &= \phi_{u2} = \phi_a, \\
i_{ua1} &= i_{ua2} = i_{ua}, \\
i_{va1} &= i_{va2} = i_{va}.
\end{aligned}$$

The corresponding features, $\mathbf{H}_{2a}$ and $\mathbf{H}_{2b}$, are computed by projecting $\mathbf{H}_{1a}$ and $\mathbf{H}_{1b}$ to image b,

$$\begin{aligned}
\mathbf{H}_{ab} &= \mathbf{H}_a^{-1}\mathbf{H}_b, \\
\mathbf{H}_{b1} &= \mathbf{H}_{ab}\mathbf{H}_{a1}, \\
\mathbf{H}_{b2} &= \mathbf{H}_{ab}\mathbf{H}_{a2}, \\
i_{ub1} &= i_{ub2} = i_{ub}, \\
i_{vb1} &= i_{vb2} = i_{vb}.
\end{aligned}$$

Here $\mathbf{H}_{ab}$ is the direct transformation mapping feature a to feature b.

7.1.3 Feature Replication

The seed region alignment process produces a set of corresponding regions that are aligned precisely, using normalisation transformations. The feature replication stage makes use of this information to explore the local plane on which each correspondence is located, in order to find further correspondences. A seed correspondence is replicated by attempting to align neighbouring regions using the seed's normalisation transformations. If the alignment of a neighbouring region succeeds, a new correspondence is created for that region.

Figure 7.5 illustrates how different feature types are replicated. Seed features split according to Section 7.1.2, step 7 are replicated only in directions that do not result in overlap between the descendants of features split from the same feature (Figure 7.5(b)). Each seed correspondence, $\mathbf{H}_1$ and $\mathbf{H}_2$, produces eight replicated correspondences (Figure 7.5(a)). Every replicated correspondence is assigned a direction vector, $\mathbf{d} = [\ d_x \quad d_y \quad 1\ ]^\top$, with $d_x, d_y \in \{-1, 0, 1\}$.

The initial parameters of each replicated correspondence, $\mathbf{H}_{r1}$ and $\mathbf{H}_{r2}$, are computed from the seed correspondence parameters using the following procedure:

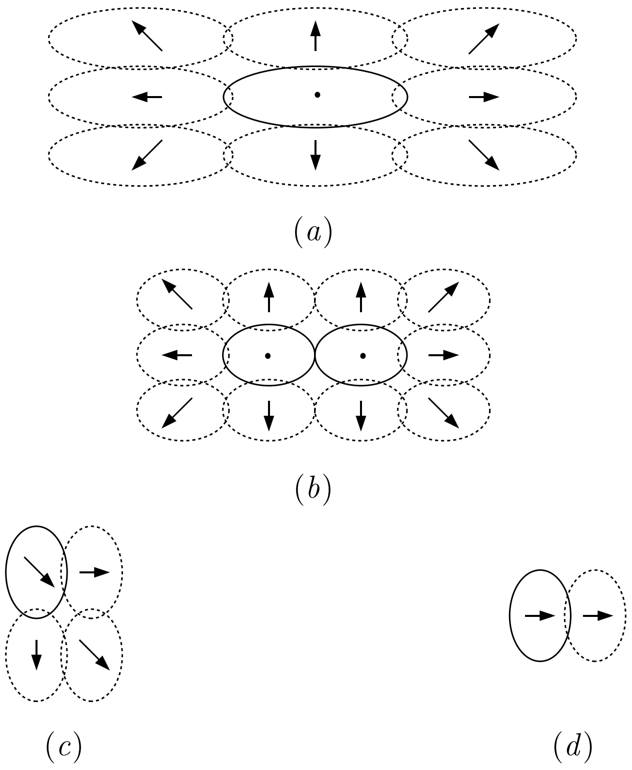


Figure 7.5: Feature replication patterns. Solid ellipses represent originating features and dashed ellipses represent replicated features. The arrow at the centre of the ellipse represents the replication direction assigned to the feature. Seed features do not have a specific replication direction and have a dot at the centre. (*a*) Seed features. (*b*) Split seed features. (*c*) Diagonally replicating features. (*d*) Horizontally or vertically propagating features.

The translation parameters of $\mathbf{H}_{r1}$ are computed using the vector $\mathbf{d}$ as,

$$\begin{aligned} t_{xr1} &= t_{x1} + 1.6k_1 \left(d_x q_1 \cos(\phi_1) - d_y q_1^{-1} \sin(\phi_1)\right), \\ t_{yr1} &= t_{ya} + 1.6k_1 \left(d_x q_1 \sin(\phi_1) + d_y q_1^{-1} \cos(\phi_1)\right). \end{aligned}$$

The remaining parameters are copied directly from $\mathbf{H}_1$. $\mathbf{H}_{r2}$ is then computed as,

$$\begin{aligned} \mathbf{H}_{12} &= \mathbf{H}_1^{-1}\mathbf{H}_2, \\ \mathbf{H}_{r2} &= \mathbf{H}_{12}\mathbf{H}_{r1}. \end{aligned}$$

The derivation of the above equations is listed in Section A.6. Each replicated correspondence is aligned (as discussed later in this section) and successfully aligned correspondences are further replicated.

Successfully replicated correspondences are further replicated based on their direction vectors. Diagonally replicated correspondences, with $d_x d_y \neq 0$, are replicated in three directions (Figure 7.5(c)):

$$\mathbf{d}_1 = \begin{bmatrix} d_x \\ d_y \\ 1 \end{bmatrix}, \quad \mathbf{d}_2 = \begin{bmatrix} 0 \\ d_y \\ 1 \end{bmatrix}, \quad \mathbf{d}_3 = \begin{bmatrix} d_x \\ 0 \\ 1 \end{bmatrix}. \tag{7.1}$$

Other correspondences (replicated horisontally and vertically) are replicated further only in the same direction (Figure 7.5(d)).

After computing the initial parameters, each new correspondence is aligned to compensate for variations in surface orientation. The alignment process is also used to validate replicated correspondences. It is expected that the alignment process will fail in the majority of cases where surface discontinuities are encountered.

The alignment procedure is similar to the seed feature alignment procedure in Section 7.1.2, with some variations. The error in replicated features is expected to be caused predominantly by variations in surface orientation between the source and replicated features, or by surface discontinuities (which result in invalid features). No initial coarse alignment process is therefore used here. It is initially unknown whether the feature image regions contain sufficient information to perform alignment – the selected region may be homogeneous. A test is performed before alignment to ensure some information is present.

The following procedure is used to align replicated features:

1. Compute normalised template and registration image patches.
2. Check that both images provide sufficient information to perform alignment.
3. Select which feature is used as the template and which feature is refined.
4. Iterative inverse compositional alignment.
5. Check that results meet expectations.

1) Normalise Images: Both features need to be normalised in order to compute the information content measure in the next step. The pair of corresponding features will be referred to as $\mathbf{H}_1$ and $\mathbf{H}_2$. The desired patch half width, $r_t^{'}$, of the normalised images is computed from the feature scales, k_1 and k_2, as,

$$r_t^{'} = \min\left\{k_1, k_2, r_t\right\},$$

where $r_t = 21$, as in Section 7.1.2. The following method is used for both features.

Given a feature $\mathbf{H}$, an affine rectified image is computed as,

$$\begin{aligned} \mathbf{H}_n &= \mathbf{T}\left(t_x, t_y\right)\mathbf{A}\left(q, \phi\right)\mathbf{R}\left(\theta\right), \\ I_a\left(\mathbf{x}\right) &= I\left(\mathbf{H}_n\mathbf{x}\right). \end{aligned}$$

where $\mathbf{H}_n$ is the same as $\mathbf{H}$ without a scaling component and $I_a(\mathbf{x})$ is computed over the coordinate range, $x, y \in k[-1, 1]$ (the local extent of the feature). Apply a Gaussian anti-aliasing filter with scale parameter, $\sigma = k/3r_t$, to image I_a. The scaled image is then computed as,

$$I_s(\mathbf{x}) = I_a\left(\mathbf{K}\left(\frac{r_t}{k}\right)\mathbf{x}\right),$$

and the final normalised image is computed as,

$$I_n = i_u I_s + i_v.$$

2) Check Information Requirement: Before attempting to align the normalised images, it is first determined whether they contain sufficient information to avoid the aperture problem. This is not necessary when aligning the original matched features (Section 7.1.2), since these features were specifically selected to be suitable for matching and alignment. In the case of replicated features, nothing is yet known about intensity distribution in the region.

The information test computes a normalised histogram of image gradients, $h(i)$, with eight direction bins, $\theta_h(i)$, $i \in \mathbb{Z}[0, 8]$. Two measures are computed from the histogram to determine if the image contains sufficient information. The first is the total energy,

$$e_h = \sum_{\forall i} h(i).$$

The second measure, d_h, is used to detect when the image contains only a single dominant edge. It is computed as the ratio of the magnitude of the mean gradient vector to the total energy.

$$\begin{aligned} \mathbf{g} = g\angle\theta_g &= \sum_{\forall i} h(i)\angle\theta_h(i), \\ d_h &= \frac{g}{e_h}. \end{aligned}$$

If both measures satisfy the thresholds,

$$\begin{aligned} e_h &> 20, \\ d_h &< 0.8, \end{aligned}$$

then the process proceeds to the next step. Otherwise, the feature is discarded. These thresholds were chosen, through trials, to simultaneously maximise the number of output features and the proportion of inlier correspondences found during the computation of the epipolar geometry of various scenes. It might be more effective to compute an appropriate threshold based on image information content – this is a topic for future research.

3) Select Template: The most eccentric feature is selected as template feature and its normalised image (computed in step 1 above) is used as the template image. The other feature is refined in the following step and its source image is used as the registration image.

4) Iterative Alignment: The inverse compositional alignment algorithm (see Section A.2 and [3]) is used to refine the registration feature.

5) Check Outcome: A mean squared error threshold of 40 and a minimum q threshold of 0.05 are applied to determine if the feature was successfully aligned. Features that do not meet these requirements are discarded.

Figure 7.6 shows a simple example where the feature replication algorithm was applied. In this case only a few unique features were correctly matched using wide baseline matching. Unfortunately, the wealth of texture information in the images is too ambiguous to be matched directly. The original wide baseline matches contained only 11 correct matches. Although this is sufficient to compute the epipolar geometry, the correct matches are buried in 80% incorrect matches, which can confuse even the most robust methods.

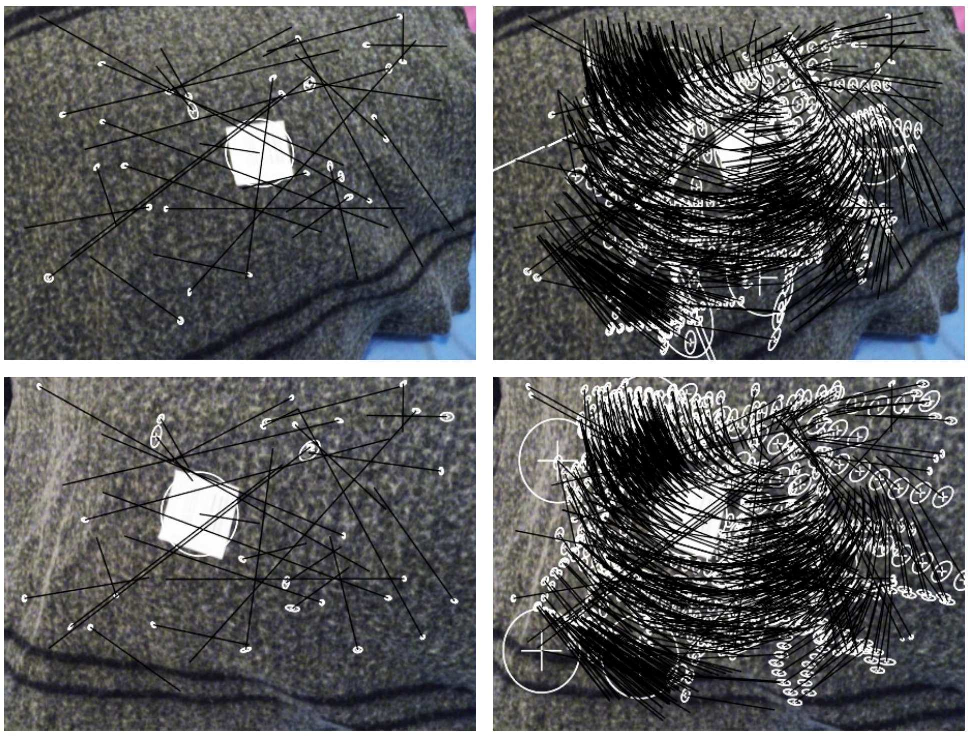


Figure 7.6: The feature replication algorithm applied to a textured, curved surface. (*a*) Input seed features and matching lines from phase 2. (*b*) Output replicated correspondences

After seed feature alignment (Figure 7.6(*a*)), the situation is somewhat improved. Using feature replication, the few correct seeds are multiplied into a large number of correct matches (Figure 7.6(*b*)). The incorrect seeds simply fail to produce more than a few replicated features. The result is a total of 790 accurate correspondences and only 6% incorrect matches after replication.

7.1.4 Dense Correspondence Extraction

The dense correspondence extraction phase attempts to find all the small scale correspondences within each aligned affine region. The iterative alignment performed in preceding phases produces a set of reasonably accurately aligned correspondences. This alignment greatly simplifies the process of finding small scale correspondences, however, it can not be assumed that every pixel in the aligned regions correspond exactly. The affine model used to align image regions does not account for surfaces that are not perfectly flat, nor for projective deformation. It is assumed that the alignment error of any point in aligned images is in the order of a few pixels.

The aperture problem plays a very significant role when attempting to align small image regions. The image patch used for alignment must contain sufficient information to determine the relationship between images unambiguously. This problem is addressed in two ways by the dense correspondence extraction phase.

Firstly, a two parameter translation transformation is used to perform the alignment, instead of a six parameter affine model. This reduces the amount of information required for unambiguous alignment. The alignment efforts of previous stages have taken care of the affine and photometric alignment sufficiently, so that it does not need to be further refined. Furthermore, only point correspondences are desired at this stage – the affine transformation between regions will not be used any further. It is therefore only necessary to find the translation between corresponding points in the normalised images.

Secondly, the alignment is only performed on regions considered to have sufficient information to determine the translation unambiguously. The determinant of Hessian [6] corner detector is used in this algorithm to select points suitable for alignment. This detector selects points that are located in areas with significant image gradients in at least two directions. This provides sufficient information to determine the translation transformation.

The dense correspondence extraction procedure is as follows:

1. Normalise image regions.
2. Extract corners.
3. Align corners.
4. Project new correspondences back to original images.
5. Remove duplicates.

Figure 7.7 demonstrates the first three steps of this process. Observe that, because the surface in these images is not flat, there is some alignment error between the images.

1) Normalise Images: The pair of corresponding features are labelled $\mathbf{H}_a$ and $\mathbf{H}_b$, such that $k_a > k_b$. The normalisation transformations is computed as,

$$\begin{aligned}\mathbf{H}_{na} &= \mathbf{T}\left(t_{xa}, t_{ya}\right)\mathbf{A}\left(q_a, \phi\right)\mathbf{R}\left(\theta_a\right),\\ \mathbf{H}_{nb} &= \mathbf{T}\left(t_{xb}, t_{yb}\right)\mathbf{A}\left(q_b, \phi\right)\mathbf{R}\left(\theta_b\right).\end{aligned}$$

Normalise image b directly,

$$I_{bn} = I_b\left(\mathbf{H}_{nb}\mathbf{x}\right), \quad x, y \in \left[-k_b, k_b\right].$$

Compute the affine-normalised image of image a as,

$$I_{aa} = I_a\left(\mathbf{H}_{na}\mathbf{x}\right), \quad x, y \in \left[-k_a, k_a\right],$$

filter I_{aa} with a Gaussian anti-aliasing filter with scale parameter $\sigma = k_b/3k_a$, and compute the normalised image a as,

$$I_{an} = I_{aa}\left(\mathbf{K}\left(\frac{k_b}{k_a}\right)\mathbf{x}\right), \quad x, y \in \left[-k_b, k_b\right].$$

This produces two normalised images for the corresponding features that are at the highest common resolution.

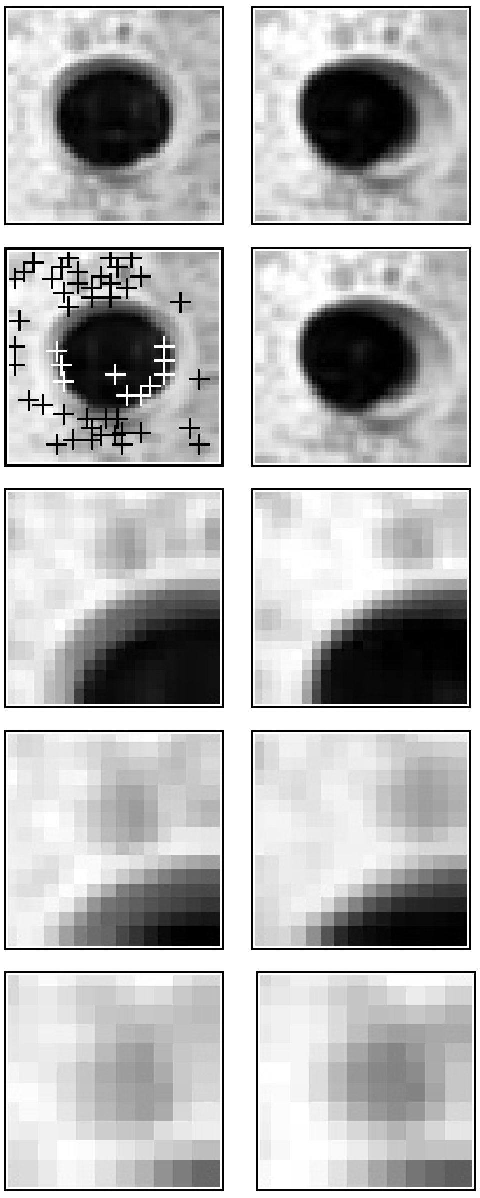


Figure 7.7: The point registration process. Each column shows one of a pair of corresponding features from two different images. Row 1 – Normalised pair of image patches (step 1). Row 2 – corner points detected in image 1, indicated as black and white crosses (step 2). Row 3 - 5 – progressively smaller aligned subregions around one of the points (step 3). Note the decreasing alignment error as the region size is reduced.

2) Extract Corners: The objective of the corner extraction step is to find points suitable for further alignment, not for matching across images. Corners are only extracted in one normalised image and are initially assumed to be at the same location in the corresponding normalised image. Corners are extracted by applying the determinant of Hessian operator to the image (at a single scale) and extracting the maxima. The determinant of Hessian of an image is computed as,

$$\begin{aligned} g_1(\mathbf{x}) &= \tfrac{1}{2\pi} e^{\frac{-\mathbf{x}^\top \mathbf{x}}{2}}, \\ I_g(\mathbf{x}) &= g_1(\mathbf{x}) * I_{an}(\mathbf{x}), \\ I_{\mathcal{H}}(\mathbf{x}) &= \left| \tfrac{\partial^2 I_g(\mathbf{x})}{\partial \mathbf{x}^2} \right|. \end{aligned}$$

The positions, $c = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, of the local maxima of $I_{\mathcal{H}}$ with a value above a threshold of 100 are then extracted.

3) Align Corners: Initially the transformation relating each point correspondence is set to $\mathbf{T}_{ab}(0,0)$. Inverse compositional alignment is then used to refine this transformation using progressively smaller image regions (refer to Section A.3 for the details of the alignment equations). In the first iteration, the alignment is performed on an image region half the width of the normalised images, $w_0 = k_b$. At each following iteration, the width is reduced according to $w_{i+1} = w_i - \lfloor \sqrt{w_i} \rfloor$. The process concludes when $w < 10$. Regions smaller than this threshold contain too few pixels for the inverse compositional alignment to compute accurate parameter updates. A region width of 10 also coincides roughly with the support region of the determinant of Hessian corners. Because there is already little error in the alignment, the inverse compositional alignment usually requires only $1 - 3$ iterations for every image size.

Figure 7.7 shows this process in action. Initially the high contrast objects in the images dominate the alignment process, resulting in alignment error at the point of interest. As the alignment region size is reduced, the high contrast interference is removed, allowing the point of interest to be aligned exactly. If the alignment process used only the smallest image size, then the initial alignment error would have been comparable to the image size and would not have been resolved.

4) Project Back: After aligning a point $\mathbf{x}_i$ and finding $\mathbf{T}_{ab}(t_x, t_y)$, the point is projected back to the original images,

$$\begin{aligned}\mathbf{x}_a &= \mathbf{H}_{na}\mathbf{K}\left(\frac{k_b}{k_a}\right)\mathbf{x}_i,\\ \mathbf{x}_b &= \mathbf{H}_{nb}\mathbf{T}_{ab}(t_x, t_y)\,\mathbf{x}_i.\end{aligned}$$

5) Remove Duplicates: During the feature replication stage (Section 7.1.3), it is possible that different seed features can produce partially overlapping replicated features. This can in turn result in duplicate dense features being extracted. The presence of several duplicates of the same point correspondence is detrimental to the robust geometry estimation process. Random Sample Consensus [19], for example, may encounter degenerate configurations more frequently and require more trials to find a good solution. Duplicates are eliminated by searching for groups of correspondences that are within a distance of one pixel of each other in both images. Each group of correspondences is then replaced by one average correspondence.

Figure 7.8 shows a simple example where dense correspondences were extracted from a matched pair of affine features.

7.1.5 Example Output

Figures[2] 7.9 - 7.11 and Figures[3] 7.12 - 7.13 present two examples where the WiDense algorithm was applied to find correspondences between views and to

[2]Images sourced from `http://www.robots.ox.ac.uk/~vgg/data/valbonne/images.tar.gz`.

[3]Images sourced from `http://www.cs.unc.edu/~marc/data/castlejpg.zip`.

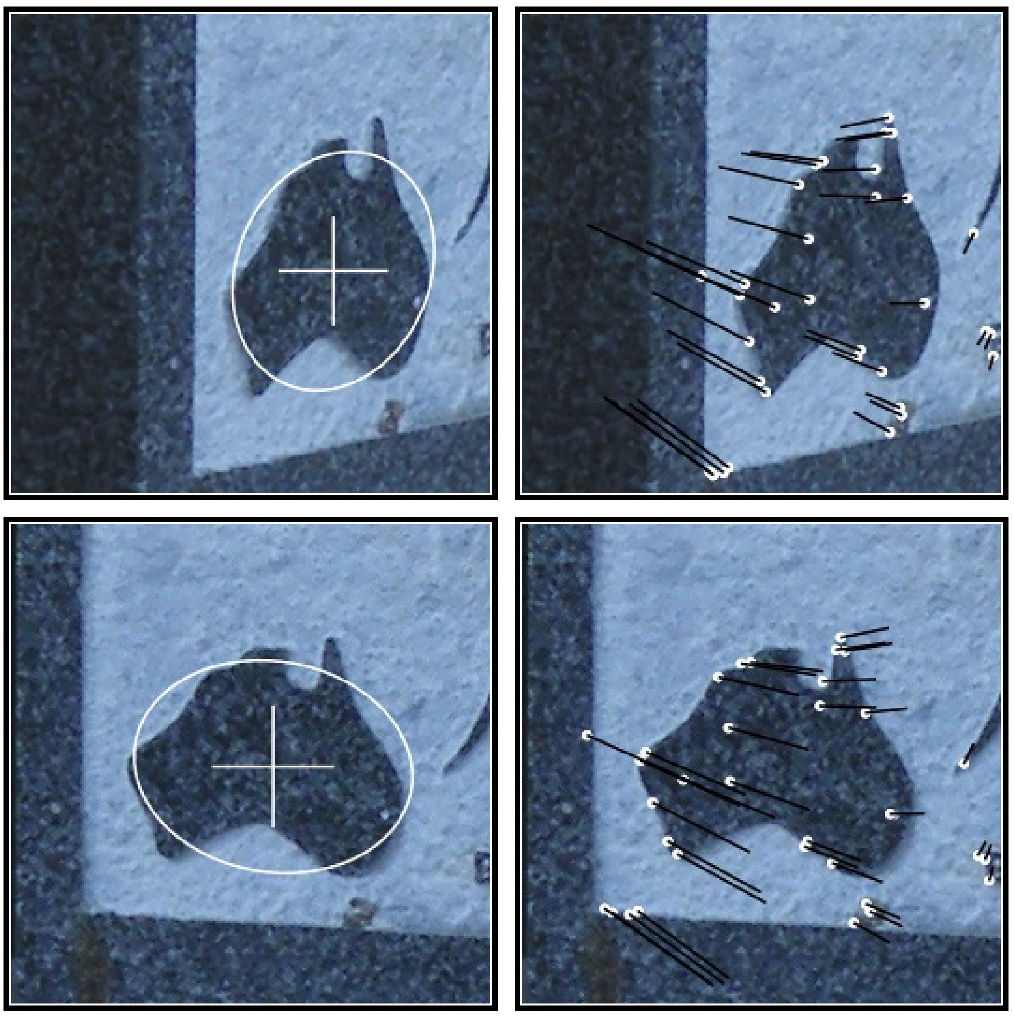


Figure 7.8: Dense correspondences extracted from a matched pair of affine features. (*a*) Input MSER features. (*b*) Dense correspondences.

compute the epipolar geometry. The wide baseline matches are in both cases not sufficient to compute the epipolar geometry accurately, while the correspondences produced by the WiDense algorithm make the computation possible and produce a large proportion of inliers.

Figures 7.9 - 7.11 shows a case where the MSER feature extractor was used (image (c)). Computing the epipolar geometry from the matched MSER features results in a grossly inaccurate, degenerate solution (image (d)). The degenerate solution is likely the result of the limited variation of depth in the scene; the MSER features are not accurate enough to resolve the depth variations.

Images (e) – (g) show the output of the three phases of the WiDense algorithm. Image (e) shows the correspondences aligned successfully during seed region alignment. It can be seen that the seed region alignment process rejects a significant proportion of incorrect matches. From the 13 seed correspondences, the replication process generates 54 correspondences (image (f)) and the dense correspondence extraction process finds 511 correspondences (image (g)). Using these correspondences, the epipolar geometry is correctly estimated (images (i) and (j)) with 81% inliers. Image (j) shows the inlier correspondences. Only 20% of the original matches were correct. The WiDense algorithm required approximately 266ms to complete on an AMD Phenom$^{\text{TM}}$ 9500 quad core PC, running in a single thread. The wide baseline matching process, by comparison, took approximately 657ms.

A second example is presented in Figures 7.12 - 7.13. Here the Hessian Affine extractor was used to find correspondences (image (c)). The epipolar geometry computation fails when only using Hessian Affine matches (not shown). Using WiDense, 189 seeds are successfully aligned, 549 regions are produced through replication, and 2131 dense correspondences are extracted. The epipolar geometry is correctly computed with 61% inliers, while only 12% of the original cor-

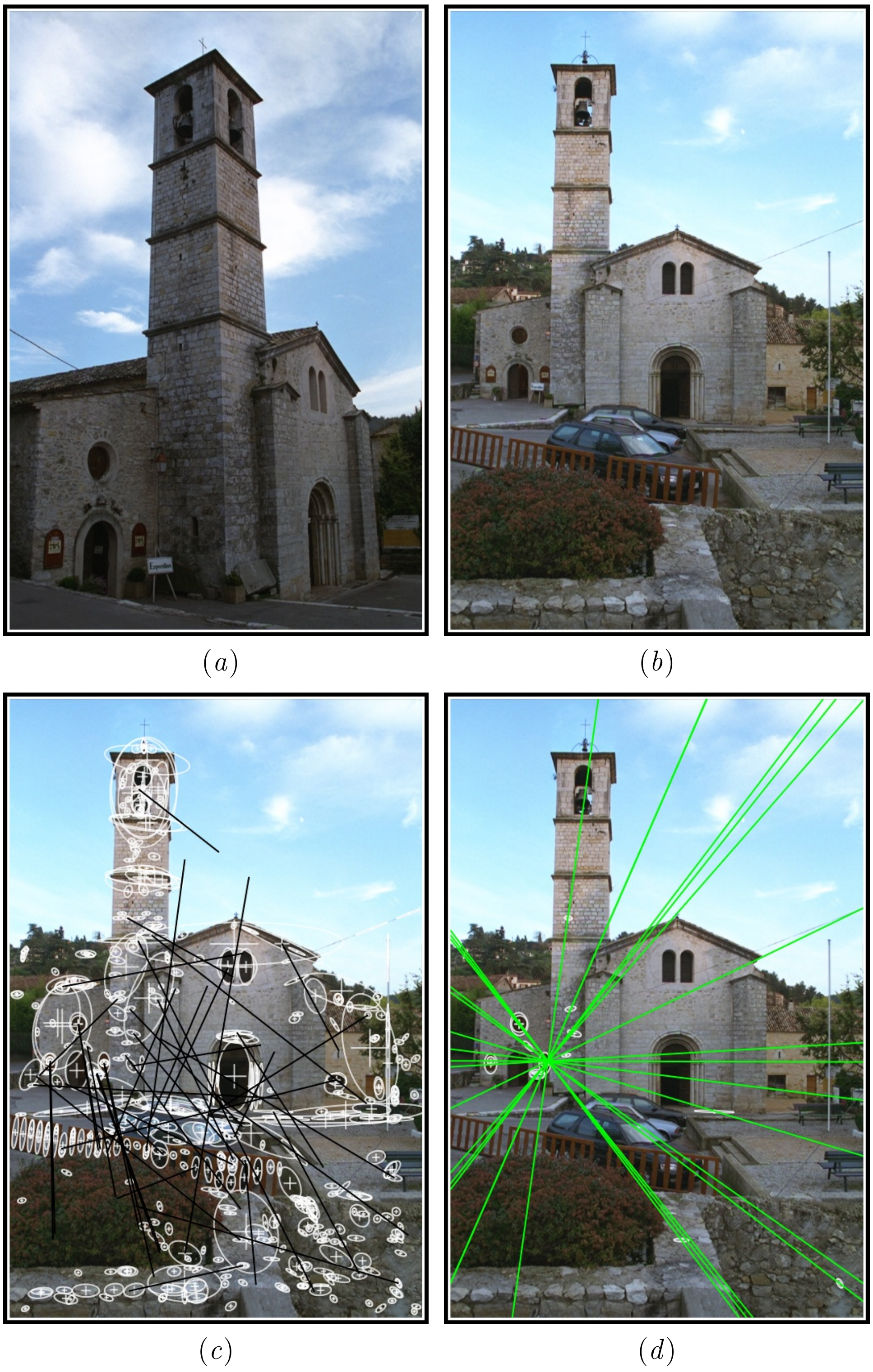
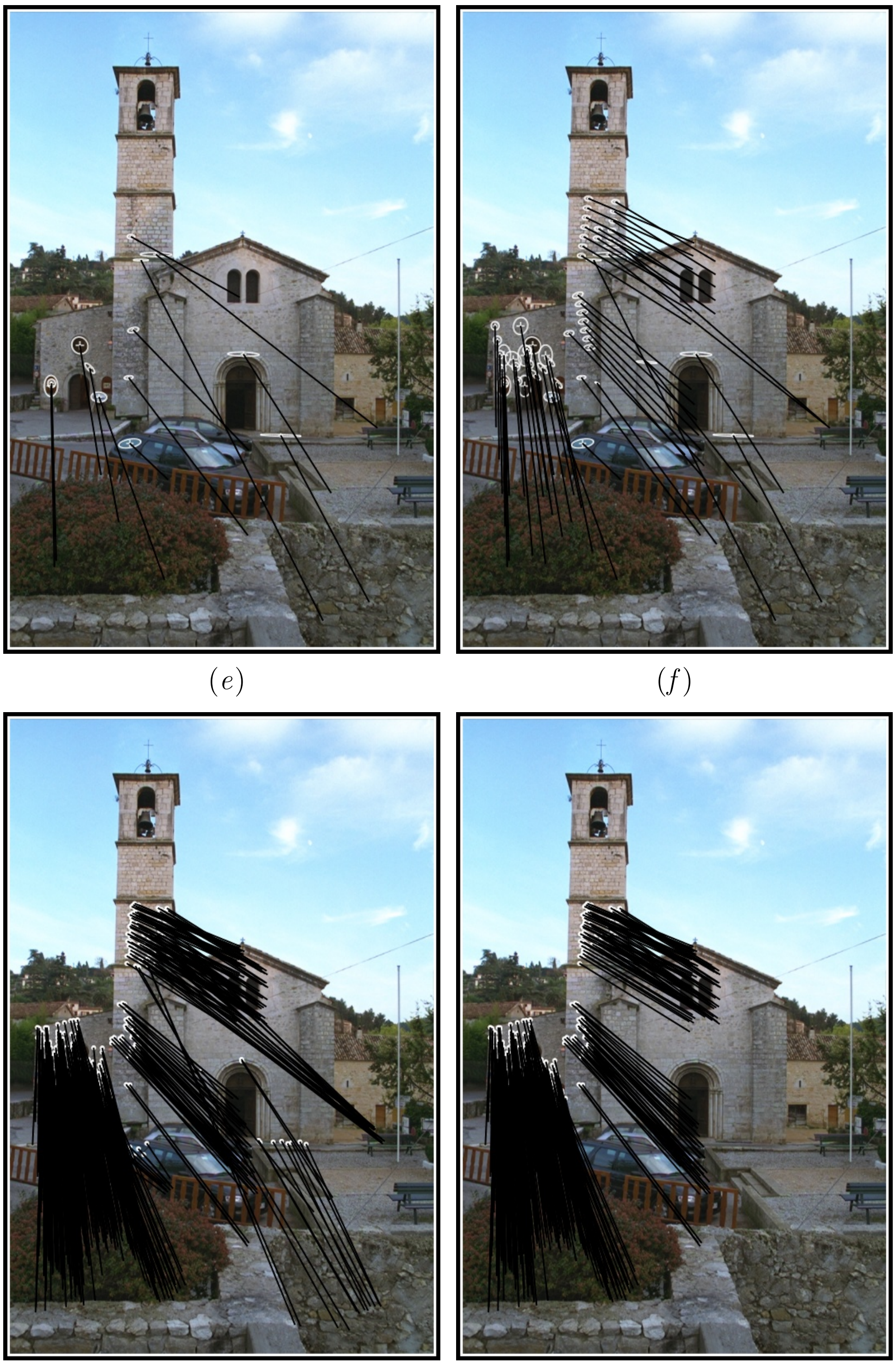


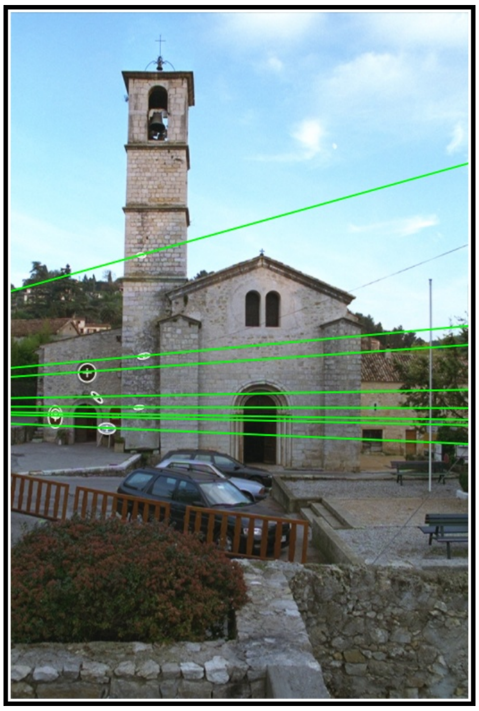
Figure 7.9: Using WiDense to compute epipolar geometry in a difficult case (continues Figure 7.10 and 7.11). (*a, b*) Input images. (*c*) MSER features and matches. (*d*) Epipolar geometry estimate (degenerate).



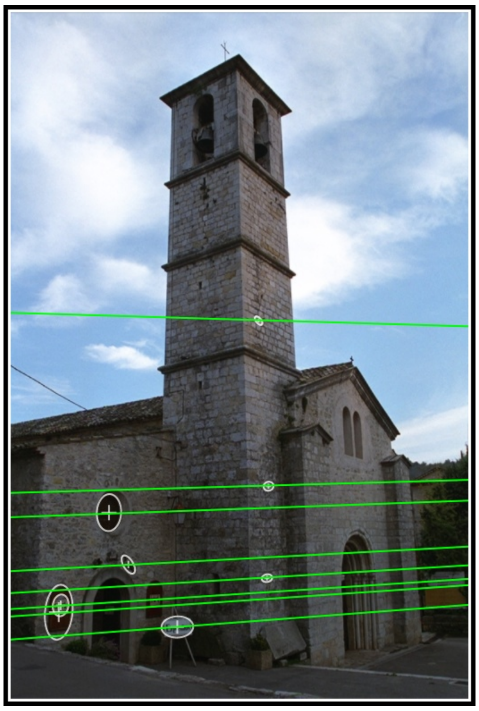
(*e*) (*f*)

(*g*) (*h*)

Figure 7.10: Continuing from Figure 7.9. (*e*) Aligned affine correspondences (phase 1). (*f*) Replicated correspondences (phase 2). (*g*) Dense correspondences (phase 3). (*h*) Inlier correspondences.



(*i*)



(*j*)

Figure 7.11: Continuing from Figure 7.10. (*i*, *j*) Epipolar geometry estimate using WiDense correspondences (correct).

respondences are correct. The WiDense algorithm completed in approximately 2.4s. This example took longer to process than the previous example due to the larger number of correspondences processed. The Hessian Affine extraction and matching took 26s.

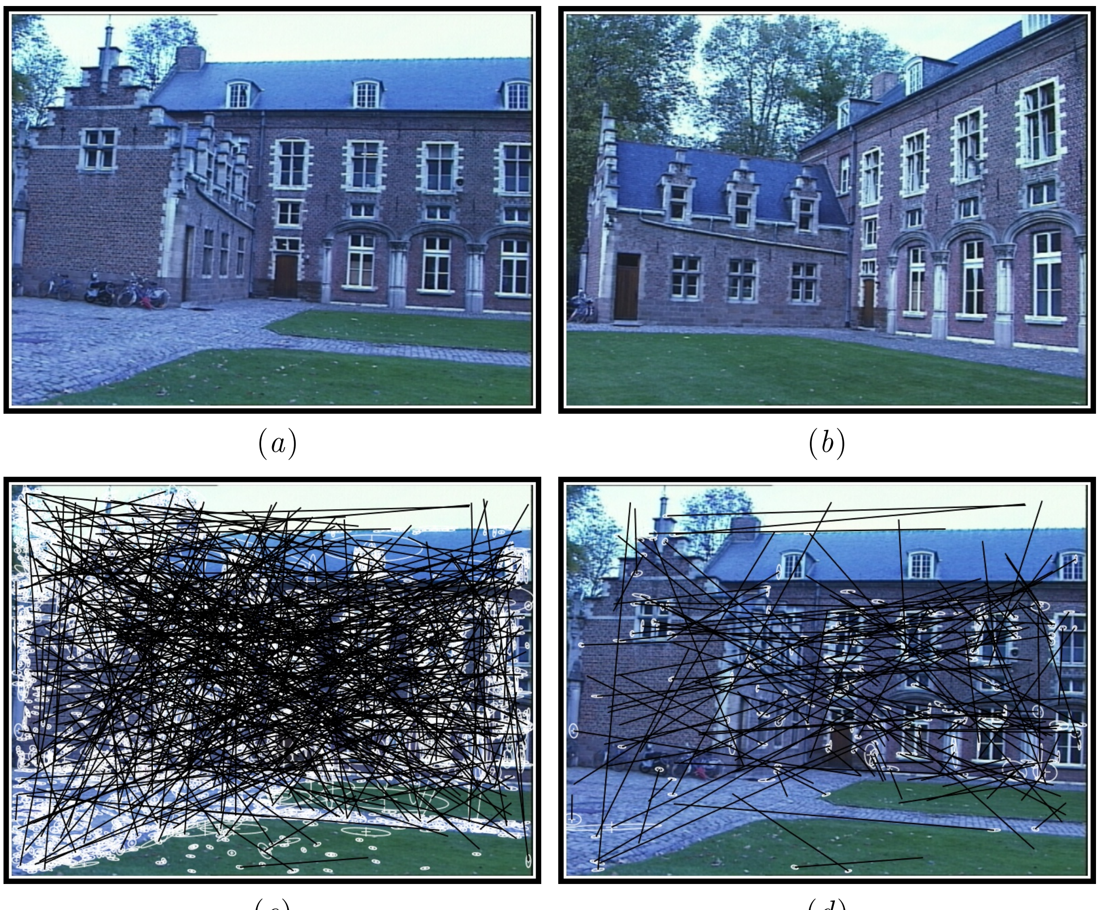
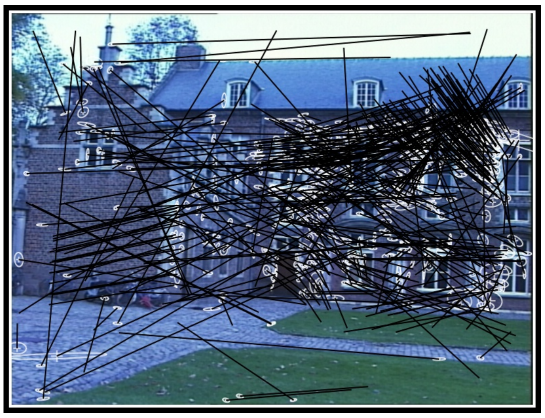
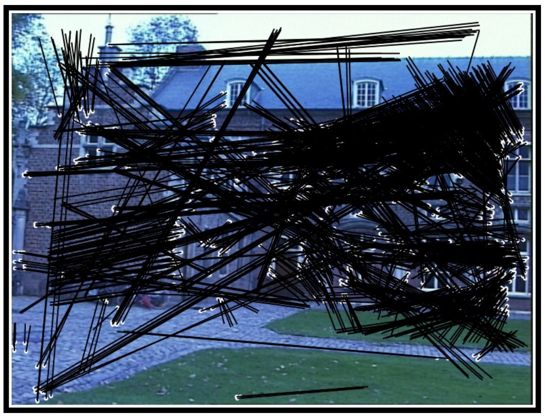
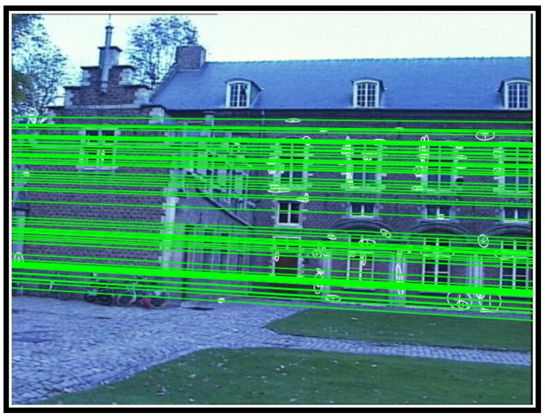


Figure 7.12: Using WiDense to compute epipolar geometry (Continues Figure 7.13). (*a*, *b*) Input images. (*c*) Hessian Affine features and putative correspondences. (*d*) Successfully aligned affine correspondences (phase 1).


(*e*)


(*f*)


(*g*)

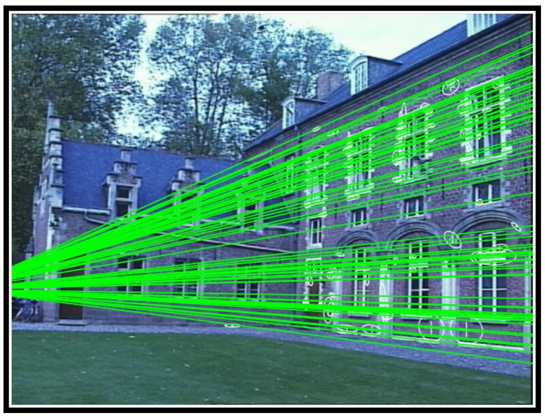

(*h*)

Figure 7.13: Continuing from Figure 7.12. (*e*) Replicated correspondences (phase 2). (*f*) Dense correspondences (phase 3). (*g, h*) Epipolar geometry estimate.

7.2 The Epipolar Geometry Computation Task

Computing the epipolar geometry is the first stage in many calibration algorithms. It can be used to constrain the search for further correspondences and can be used to generate a reconstruction of the scene and cameras with projective ambiguity [22]. The Epipolar Geometry Computation Task is an experiment that evaluates a correspondence extraction method in terms of its usefulness in computing the epipolar geometry of a set of scenes. An experiment trial consists of attempting to compute the epipolar geometry of a pair of cameras using a given correspondence extractor, and then comparing the results with ground truth data. If the epipolar geometry estimate is sufficiently accurate (its error is below a theoretical threshold), then the test trial is considered successful. The average success rate is computed over many trials and many input scenes. This test procedure evaluates how likely it is that a particular correspondence extraction technique will generate an epipolar geometry estimate that is sufficiently accurate for practical applications.

7.2.1 Data

Six sets of test data were acquired using a pair of digital cameras arranged to view a scene from widely separated views. Each set consists of images taken with the cameras in fixed position. The camera positions were varied between sets. The contents of the scene were altered for each pair of test images. Images were captured at high resolution (4.1 and 10 million pixel cameras were used) to compute the ground truth geometry. Test images are generated by scaling the original images to 640×480 pixel resolution. Scale factors are recorded for relating the test image geometry back to ground truth geometry. Scaling images to a low resolution removes a large proportion of the features from the images, ensuring

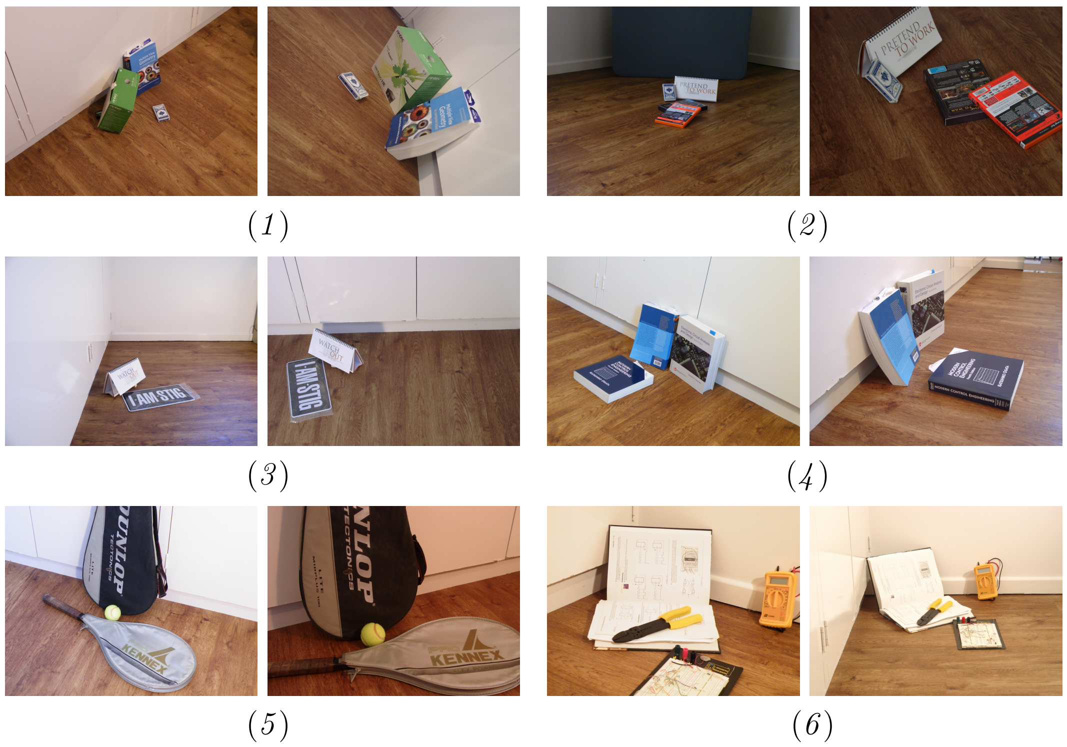


Figure 7.14: Example images from each dataset

that the task of computing the geometry of these test images is challenging. Figure 7.14 shows a pair of example images from each dataset. The dataset may easily be extended in future using the same capturing procedure listed above.

7.2.2 Generating Ground Truth Data

The ground truth data consists of a large set of accurate point correspondences for each dataset. The error in a given estimate of the epipolar geometry is measured by computing the error of the set of ground truth correspondences when compared against the estimated geometry. The ground truth data is generated automatically from the high resolution images in the dataset and then mapped to the coordinates of the test images. The following procedure is used to generate the ground truth data for each dataset:

1. Features are extracted and matched across each high resolution image pair using the MSER feature extractor. The correspondences for all the image pairs in a dataset are collected into one large set of correspondences.

2. An initial estimate of the epipolar geometry is computed using RANSAC [19] and the normalised eight point algorithm [22, 36, 37].

3. Features are matched again, using the initial geometry estimate to constrain matching so that all correspondences are inliers.

4. The WiDense algorithm is applied using the inlier correspondences as input. Because the matches are at this stage verified by means of the epipolar constraint, the WiDense algorithm produces a large number of highly accurate correspondences and few outliers.

5. A more accurate estimate of the epipolar geometry is computed from the WiDense correspondences. The dense correspondences collected from the entire dataset are used.

6. The set of inlier correspondences are scaled to match the resolution of the test images and are kept as ground truth data.

Note that the exact method used for extracting ground truth correspondences is not critical, as long as a sufficient number of accurate correspondences are generated. The quality of the ground truth correspondences is ensured by verifying all correspondences using the epipolar geometry. Because the ground truth epipolar geometry is computed using a large set of high resolution input images, it is likely that it will be of good quality.

7.2.3 Test Procedure

The test procedure is designed to measure the success rates of different correspondence extraction systems when applied to the task of estimating the epipolar geometry. Each test trial proceeds as follows:

1. Correspondences are extracted between one pair of images from the dataset using a particular extraction system.
2. The epipolar geometry of the image pair is estimated from the correspondences using RANSAC and the normalised eight point method.
3. The error of the epipolar geometry estimate is computed as the Sampson distance [22] of the ground truth correspondences. The Sampson distance is a first-order approximation to the geometric error and is defined as,

$$e_s = N_{\mathbf{x}}^{-1} \sum_{\forall \mathbf{x}} \frac{\left(\mathbf{x}'^{\top}\hat{\mathbf{F}}\mathbf{x}\right)^2}{\left(\hat{\mathbf{F}}\mathbf{x}\right)_1^2 + \left(\hat{\mathbf{F}}\mathbf{x}\right)_2^2 + \left(\hat{\mathbf{F}}^{\top}\mathbf{x}'\right)_1^2 + \left(\hat{\mathbf{F}}^{\top}\mathbf{x}'\right)_2^2},$$

   where $\mathbf{x}$ and $\mathbf{x}'$ are the ground truth correspondences, $N_{\mathbf{x}}$ is the number of correspondences, and $\hat{\mathbf{F}}$ is the epipolar geometry estimate.
4. The error, e_s, is compared to three thresholds, $t_1 = 4, t_2 = 16, t_3 = 64$. If e_s is below a threshold, the model is considered sufficiently accurate for that threshold category and the trial is considered a success. The three thresholds are used to represent the precision requirements of three hypothetical users of the geometry estimate.

One hundred trials are run for each extractor and image pair combination, and the average success rate is computed to compensate for the variability in the RANSAC method. Finally, the average number of successful trials for each dataset is reported.

7.3 Evaluation of the WiDense Algorithm

7.3.1 Correspondence Extractors Tested

The primary objective is to evaluate whether the WiDense algorithm results in improved success rates when used to supplement wide baseline matching methods. To this end, four correspondence extraction systems were implemented and tested with and without using the WiDense algorithm. The following three feature extractors were used:

1. The Harris Affine extractor using the Harris operator [21] for point extraction; the Laplacian function and the scale space primal sketch-based scale selection method (Chapter 5, Section 5.3); the second moment matrix for affine shape estimation [4, 35] (labeled HaA).
2. The Hessian Affine extractor using the determinate of the Hessian operator [6] for point extraction; the determinate of the Hessian function and scale space primal sketch-based scale selection method (Chapter 5, Section 5.3); the Hessian matrix for affine shape estimation (Chapter 6) (labeled HeA).
3. The Maximally Stable Extremal Region extractor [40] (labeled MSER).

The fourth correspondence extraction system used all three above feature extractors.

The SIFT descriptor [38, 39] was used to describe features. Features were matched using one-to-one nearest neighbour matching. A feature with two nearest neighbours at distances d_1 and d_2 with $d_1 < d_2$ is matched to the feature at distance d_1 if $d_1 < 0.5$ and $d_1/d_2 < 0.95$. The first threshold specifies the maximum matching distance and the second threshold prevents ambiguous matches [39].

set	n_f	HaA	aligned	replicated	dense	WiDense
1	18	0.00	0.00	0.00	0.00	0.00
2	19	0.00	0.00	0.00	0.00	0.37
3	39	0.71	3.00	3.00	2.96	3.45
4	20	0.00	0.00	1.00	0.00	0.00
5	46	2.60	8.19	9.82	10.35	9.56
6	41	12.54	10.30	14.00	17.90	21.45
tot.	183	15.85	21.49	27.82	31.21	34.83
%		8.66	11.74	15.20	17.05	19.03

Table 7.1: Test results for the Harris Affine extractor and the various stages of the WiDense algorithm. The average number of successful epipolar geometry computation trials with error threshold $t_2 = 16$ are listed.

7.3.2 Results

The test results for threshold level $t_2 = 16$ are presented in Tables 7.1 - 7.4. Each table presents the results for one of the feature extractors and the various stages of the WiDense algorithm. Each table entry lists the average number of image pairs for which the epipolar geometry was computed successfully. Results are included for every stage of the WiDense algorithm so that the significance of each stage may be evaluated. The “set” column lists the dataset number and the “n_f” column lists the number of frames in the corresponding dataset. The third column contains the results of using the relevant extractor alone. The “aligned”, “replicated” and “dense” columns give the results of only using the features produced by the alignment, replication and dense matching stages, respectively. In the “WiDense” column, all the features produced through the WiDense process are combined. Results for the other thresholds show mostly similar trends, and are not presented in so much detail.

Figure 7.15 presents the average results over all test data for each threshold, indicating the effect that the threshold has on success rates.

set	n_f	HeA	aligned	replicated	dense	WiDense
1	18	0.00	0.00	0.00	0.00	0.00
2	19	0.00	0.00	0.00	3.00	2.00
3	39	3.00	5.00	3.00	6.00	7.00
4	20	1.00	1.00	1.00	2.00	0.83
5	46	8.75	12.00	13.53	17.53	19.34
6	41	13.49	18.16	21.85	26.39	23.27
tot.	183	26.24	36.16	39.38	54.92	52.44
%		14.34	19.76	21.52	30.01	28.66

Table 7.2: Test results for the Hessian Affine extractor and the various stages of the WiDense algorithm. The average number of successful epipolar geometry computation trials with error threshold $t_2 = 16$ are listed.

set	n_f	MSER	aligned	replicated	dense	WiDense
1	18	0.00	2.90	6.00	6.00	5.00
2	19	0.00	1.00	2.00	5.77	4.80
3	39	8.82	10.67	8.41	13.45	11.23
4	20	4.00	1.00	1.68	5.00	5.03
5	46	12.31	19.72	18.56	25.77	25.90
6	41	16.89	19.37	22.33	25.50	24.18
tot.	183	42.02	54.66	58.98	81.49	76.14
%		22.96	29.87	32.23	44.53	41.61

Table 7.3: Test results for the MSER extractor and the various stages of the WiDense algorithm. The average number of successful epipolar geometry computation trials with error threshold $t_2 = 16$ are listed.

set	n_f	All	aligned	replicated	dense	WiDense
1	18	1.08	2.48	3.00	5.00	4.38
2	19	0.79	1.00	0.00	4.54	4.12
3	39	9.74	12.12	11.00	15.76	18.46
4	20	3.38	5.66	6.00	6.33	6.45
5	46	19.44	26.32	24.73	26.60	26.89
6	41	24.71	29.90	29.94	32.16	31.06
tot.	183	59.14	77.48	74.67	90.39	91.36
%		32.32	42.34	40.80	49.39	49.92

Table 7.4: Test results for all the extractors combined and the various stages of the WiDense algorithm. The average number of successful epipolar geometry computation trials with error threshold $t_2 = 16$ are listed.

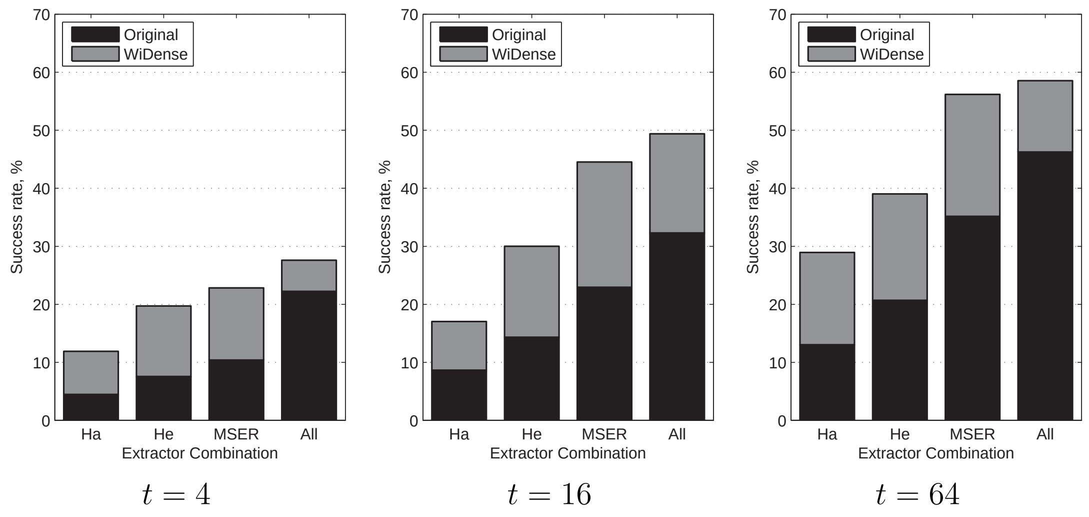


Figure 7.15: Success rate results averaged over all datasets for each of the three error thresholds

7.3.3 Discussion

The results show a large range of variation across the different correspondence extraction systems. The difference between the Harris, Hessian and MSER-based systems is consistent with previous evaluations of these feature extractors [45, 47].

The difference in results between different data sets is the result of different camera geometry. Some camera configurations result in more difficult scenarios for geometry computation than others.

The addition of WiDense consistently improves the results for every correspondence extractor. An increase of between 1.3 and 2.3 times the success rate was observed. It can be seen from Figure 7.15 that the WiDense algorithm provides a consistent performance improvement over a range of accuracy requirements.

The alignment stage improves the success rate significantly by itself, even though it was originally intended to prepare the input features for replication and dense feature extraction stages. The replication stage makes a small improvement to the success rate on average. Its main contribution is a larger set of regions to search for dense features. The dense extraction stage gives the most significant increase in success rate.

There is little difference between tests using only the dense correspondences and dense and replicated correspondences combined ("WiDense" column). This is likely due to the fact that the dense correspondences are more accurate than the features from previous stages and account for a dominant proportion of the total feature set. It appears that it is most beneficial to only use the dense correspondences for computing the scene and camera geometry. These features are also the closest to ideal point correspondences and may be more useful in the reconstruction process than the original affine features with their large support regions.

7.4 Chapter Summary

The Uncalibrated Wide Baseline Dense Optical Flow algorithm, or WiDense algorithm, and the Epipolar Geometry Estimation Task are presented in this chapter. The WiDense algorithm makes use of the information contained in a set of putative wide baseline matches to find a large number of accurate correspondences between two views. It consists of three phases. The first phase accurately aligns the support regions of each input correspondence and removes a significant proportion of incorrect matches in the process. The second phase explores the neighbourhood of each correspondence and aligns many additional correspondences. In the last phase, fine scale correspondences are extracted from each aligned correspondence, yielding a large set of accurate point correspondences.

The WiDense algorithm is useful for extracting additional correspondences in difficult scenarios where even the best wide baseline matching methods cannot produce sufficient accurate correspondences to compute the camera geometry. It is particularly useful in highly textured scenes, scenes with many similar features that are too ambiguous to match reliably using other methods, and scenes with little variation in depth, where high accuracy is required to avoid getting stuck on a degenerate solution. WiDense can produce large numbers of additional correspondences and requires processing time on par with current wide baseline matching algorithms.

Experiments have demonstrated that the WiDense algorithm can assist in computing the epipolar geometry in challenging scenarios. Using the WiDense algorithm, in addition to an established feature extractor, allows successful computation of the epipolar geometry in, on average, twice as many cases as using the feature extractor alone. These results indicate that the WiDense algorithm is very useful for finding correspondences and computing the geometry of complex scenes.

The Epipolar Geometry Estimation Task is a method for evaluating how useful correspondence extraction methods are for computing the epipolar geometry. It provides a way to compare correspondence extractors in terms of their utility in a real world calibration task.

Chapter 8

Conclusions and Future Work

The ability to calibrate cameras automatically is a critical requirement for multi-view computer vision algorithms to be deployed to real surveillance networks on a large scale. The process of finding correspondences between overlapping views is a cornerstone of automatic camera calibration. This task is challenging in cases where cameras are sparsely arranged (where the baseline between cameras is relatively wide).

The research presented in this thesis makes contributions in two primary research directions. The first is the further development and improvement of robust feature extraction algorithms. Outcomes from this research direction are discussed in Section 8.1. The second is the exploration of methods beyond the traditional feature matching approach. New techniques are presented that explore the scene further, based on the information produced through matching. These techniques are referred to as second tier correspondence extraction techniques. The outcomes of this research direction are discussed in Section 8.2. Ideas for future research directions in the wide baseline matching field are discussed in Section 8.3.

8.1 Salient Feature Research

The mathematics used to express and solve the problems addressed in this thesis are introduced in Chapter 2. The problem of wide baseline matching is presented in Chapter 3, followed by a review of the feature extraction and matching literature. Chapter 4 presents a review of saliency map-based feature extraction techniques. An algorithmic framework for saliency map-based affine covariant feature extraction is presented. This framework is used to compare the performance of the new methods developed in later chapters.

In Chapter 5, a new method for scale space analysis of local features is presented. The scale space primal sketch is adapted to the scale space feature sketch (SSFS). An algorithm is devised to extract the sketch from a set of multi-scale features. This algorithm operates on the features only, and does not require any further image processing. It therefore requires little processing time and does not place any requirements on how the scale space pyramid is constructed. A scale selection method based on the SSFS is presented. This novel scale selection method is more effective, accurate and efficient than existing methods. These benefits are most pronounced when using different functions for spatial position selection and scale selection, or when using affine adaptation.

Chapter 6 demonstrates that the Hessian matrix can be used to estimate the shape of local image features in the same way the second moment matrix is commonly used. The Hessian matrix is simpler to compute and delivers a significant gain in extractor efficiency, in comparison to the second moment matrix. Using the Hessian matrix to estimate the shape of blob-like features results in features of equivalent quality to using the second moment matrix. Applying the Hessian matrix to corner features, however, results in a significant reduction in feature quality.

8.2 Second Tier Correspondence Extraction Techniques

The wide baseline matching approach can only produce a limited number of features, due to the amount of information required to describe each feature. This means that each matched feature contains a significant amount of information in the form of normalisation transformations and local image gradients. Existing methods treat matched affine features as point correspondences and do not take advantage of this information. Chapter 7 presents a set of novel procedures for extracting the information contained in matched features and utilising it to find additional correspondences. These procedures are combined to form the Uncalibrated Wide Baseline Dense Optical Flow algorithm, called WiDense.

The WiDense algorithm consists of four phases. The first phase is the conventional wide baseline matching step. In the second phase, the affine feature matches produced by the first phase are accurately aligned. During this phase, many incorrect matches are detected and discarded. The third phase makes use of the feature normalisation transformations to search the neighbourhood of each match for more regions that can be aligned. The fourth phase selects many small scale features from each aligned pair of features, and aligns them with high accuracy. During the third and fourth stages, correct matches produce many replicated and dense correspondences, while incorrect matches tend to produce few additional incorrect correspondences. The result is a large set of correspondences, of which the majority are highly accurate.

Chapter 7 presents a method for evaluating wide baseline correspondence extraction systems. The evaluation system operates by attempting to compute the epipolar geometry of a set of scenes using the matches produced by a given wide baseline matching system. By means of this evaluation, it is found that the WiDense system achieves a success rate of between 1.3 and 2.3 times that of existing methods.

8.3 Future Work

The local feature matching approach is reaching maturity. It is expected that further research into feature extractors will only yield incremental improvements over existing methods. This thesis recognises this limitation and has begun to explore techniques that go beyond feature extraction and matching. The WiDense system represents the first major step towards developing a paradigm of second tier correspondence extraction methods.

Each of the four phases of the WiDense system represents a major approach to extracting information from a set of affine matches. Possibilities for future research exist within each phase. The feature alignment stage could be included in the matching process directly, to help identify incorrect matches. For example, by applying the alignment step to the top two or three nearest neighbour matches and observing the final alignment error, it may be possible to discern between features that appear ambiguous in descriptor space. The alignment system currently makes use of a fixed threshold to select between successfully and unsuccessfully aligned matches. Ideally this should be replaced by a system that selects the threshold based on the image information content, thereby improving the utility of this step in discerning correct and incorrect matches.

The replication stage could be improved by using more intelligent search strategies and methods that are robust to small occlusions. Man-made environments frequently contain surfaces that are coplanar but disjoint. A strategy for searching the entire image for surfaces that are coplanar to a given feature may be able to find many additional correspondences. Detecting duplicate and overlapping regions in this phase could help reduce redundant processing.

The WiDense system as a whole could be extended to dealing with more than two overlapping views at a time. Techniques such as congealing [11] could be used to simultaneously align features across many views. Future research may also attempt to find entirely new ways to exploit the information provided by matched affine features.

The development of wide baseline matching systems is hampered by the lack of data suitable for supporting algorithm development and evaluation. The ability to generate test cases from artificial 3D data would be very useful to research in this field. Computer graphics could be used to generate images from 3D models. Example sources of useful data include CAD models produced by architects, or the maps used in game engines. This approach would make it possible to generate test data with highly accurate ground truth information (full camera and scene geometry) and precisely controlled variations in viewing angle, view overlap, baseline, illumination, camera exposure settings, lens configuration, and many more.

Bibliography

[1] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda, "Uniqueness of the gaussian kernel for scale-space filtering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 8, no. 1, pp. 26–33, 1986.

[2] B. Babenko, P. Dollár, and S. Belongie, "Task specific local region matching," in *Proc. International Conference on Computer Vision*, 2007.

[3] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework: Part 1," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[4] A. Baumberg, "Reliable feature matching across widely separated views," in *Proc. Computer Vision and Pattern Recognition, IEEE Conference on*, vol. 1, (Hilton Head Island, South Carolina, USA), pp. 774–781, 2000.

[5] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proc. European Conference on Computer Vision*, (Graz, Austria), pp. 404–417, 2006.

[6] P. Beaudet, "Rotationally invariant image operators," in *Proc. Pattern Recognition, International Joint Conference on*, (Kyoto, Japan), pp. 579–583, 1978.

[7] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 4, pp. 509–522, 2002.

[8] D. Buesching, "Efficiently finding bitangents," in *Proc. Pattern Recognition, International Conference on*, vol. 1, pp. 428–432 vol.1, 1996.

[9] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[10] M. Couprie, L. Najman, and G. Bertrand, "Quasi-linear algorithms for the topological watershed," *Journal of Mathematical Imaging and Vision*, vol. 22, no. 2-3 / May, 2005, pp. 231–249, 2005.

[11] M. Cox, S. Lucey, S. Sridharan, and J. Cohn, "Least-squares congealing for large numbers of images," in *In proceedings: International Conference on Computer Vision*, 2009.

[12] J. Crowley and A. Parker, "Representation for shape based on peaks and ridges in the difference of low-pass transform," Tech. Rep. CMU-RI-TR-83-04, Carnegie Mellon University, May, 1983 1983.

[13] F. Dellaert, S. M. Seitz, S. Thrun, and C. Thorpe, "Feature correspondence: A markov chain monte carlo approach," in *Proc. Advances in Neural Information Processing Systems*, 2001.

[14] R. Deriche, "Recursively implementing the gaussian and its derivatives," tech. rep., INRIA, April 1993.

[15] M. Donoser and H. Bischof, "Efficient maximally stable extremal region (mser) tracking," in *Proc. Computer Vision and Pattern Recognition, IEEE Conference on*, vol. 1, pp. 553–560, IEEE, 2006.

[16] Y. Dufournaud, C. Schmid, and R. Horaud, “Image matching with scale adjustment,” *Computer Vision and Image Understanding*, vol. 93, no. 2, pp. 175–194, 2004.

[17] S. El Mejdani, R. Egli, and F. Dubeau, “Old and new straight-line detectors: Description and comparison,” *Pattern Recognition*, vol. 41, no. 6, pp. 1845–1866, 2008.

[18] V. Ferrari, T. Tuytelaars, and L. Van Gool, “Wide-baseline multiple-view correspondences,” in *Proc. Computer Vision and Pattern Recognition*, vol. 1, pp. 718–728, IEEE Computer Society; 1999, 2003.

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

[20] W. Freeman and E. Adelson, “The design and use of steerable filters,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 13, no. 9, pp. 891–906, 1991.

[21] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proc. Alvey Vision Conference*, pp. 189–192, 1988.

[22] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* New York: Cambridge University Press, 2 ed., 2003.

[23] A. E. Johnson and M. Hebert, “Recognizing objects by matching oriented points,” in *Proc. Computer Vision and Pattern Recognition*, pp. 684–689, 1997.

[24] T. Kadir and J. Brady, “Scale, saliency and image description,” *International Journal of Computer Vision*, vol. 45, no. 2, pp. 83–105, 2001.

[25] T. Kadir, A. Zisserman, and M. Brady, “An affine invariant salient region detector,” in *Proc. European Conference on Computer Vision*, pp. 228 – 241, 2004.

[26] Y. Ke and R. Sukthankar, “Pca-sift: a more distinctive representation for local image descriptors,” in *Proc. Computer Vision and Pattern Recognition*, vol. 2, pp. 511 – 517, 2004.

[27] J. Koenderink and A. van Doorn, “Representation of local geometry in the visual system,” *Biological Cybernetics*, vol. 55, no. 6, pp. 367–375, 1987.

[28] F. Kristensen and W. MacLean, “Real-time extraction of maximally stable extremal regions on an fpga,” in *Proc. Circuits and Systems, International Symposium on*, pp. 165–168, 2007.

[29] S. Lazebnik, C. Schmid, and J. Ponce, “Sparse texture representation using affine-invariant neighborhoods,” in *Proc. Computer Vision and Pattern Recognition*, pp. 319–324, 2003.

[30] S. Lazebnik, C. Schmid, and J. Ponce, “A sparse texture representation using local affine regions,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1265–1278, 2005.

[31] T. Lindeberg, *Discrete Scale-Space Theory and the Scale-Space Primal Sketch.* PhD thesis, Royal Institute of Technology, 1991.

[32] T. Lindeberg, “Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention,” *International Journal of Computer Vision*, vol. 11, no. 3, pp. 283–318, 1993.

[33] T. Lindeberg, *Scale-Space Theory in Computer Vision.* Boston: Kluwer Academic, 1st ed., 1994.

[34] T. Lindeberg, “Feature detection with automatic scale selection,” *International Journal of Computer Vision*, vol. 30, no. 2, pp. 77–116, 1998.

[35] T. Lindeberg and J. Gårding, “Shape-adapted smoothing in estimation of 3-d depth cues from affine distortions of local 2-d brightness structure,” in *Proc. European Conference on Computer Vision*, vol. 800/1994 of *Lecture Notes in Computer Science*, pp. 389–400, Berlin / Heidelberg: Springer, 1994.

[36] H. C. Longuet-Higgins, “A computer algorithm for reconstructing a scene from two projections,” *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[37] H. C. Longuet-Higgins, “A computer algorithm for reconstructing a scene from two projections,” in *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, p. 61, San Francisco: Morgan Kaufmann Publishers Inc., 1987.

[38] D. Lowe, “Object recognition from local scale-invariant features,” in *Proc. International Conference on Computer Vision*, (Corfu, Greece), pp. 1150–1157, 1999.

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

[40] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide baseline stereo from maximally stable extremal regions,” in *Proc. British Machine Vision Conference* (P. R. a. D.Marshall, ed.), vol. 1, (Cardiff, UK), pp. 384–393, 2002.

[41] K. Mikolajczyk and J. Matas, “Improving descriptors for fast tree matching by optimal linear projection,” in *Proc. International Conference on Computer Vision*, pp. 1–8, 2007.

[42] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," in *Proc. Computer Vision and Pattern Recognition*, vol. 2, pp. 257–263, 2003.

[43] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, pp. 63–86, 2004.

[44] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[45] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *International Journal of Computer Vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[46] F. Mindru, T. Moons, and L. Van Gool, "Recognizing color patterns irrespective of viewpoint and illumination," in *Proc. Computer Vision and Pattern Recognition*, vol. 1, pp. 368–373, 1999.

[47] P. Moreels and P. Perona, "Evaluation of features detectors and descriptors based on 3d objects," *International Journal of Computer Vision*, vol. 73, no. 3, pp. 263–284, 2007.

[48] M. Perdoch, J. Matas, and S. Obdrzalek, "Stable affine frames on isophotes," in *Proc. Computer Vision, International Conference on*, pp. 1–8, 2007.

[49] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. European Conference on Computer Vision*, vol. 3951/2006 of *Lecture Notes in Computer Science*, (Berlin / Heidelberg), pp. 430–443, Springer, 2006.

[50] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American Statistical Association*, vol. 79, no. 388, pp. 871–880, 1984.

[51] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets," in *Proc. Computer Vision, European Conference on*, Lecture Notes in Computer Science, pp. 414–431, Copenhagen, Denmark: Springer Berlin / Heidelberg, 2002.

[52] C. Schmid, R. Mohr, and C. Bauckhage, "Comparing and evaluating interest points," in *Computer Vision, 1998. Sixth International Conference on*, pp. 230–235, 1998.

[53] J. Sporring, L. Florack, M. Nielsen, and P. Johansen, *Gaussian scale-space theory.* Kluwer Academic Publishers Norwell, MA, USA, 1997.

[54] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2008.

[55] T. Tuytelaars and L. Van Gool, "Wide baseline stereo matching based on local, affinely invariant regions," in *Proc. British Machine Vision Conference*, (Bristol), pp. 412–425, 2000.

[56] T. Tuytelaars, L. Van Gool, L. D'haene, and R. Koch, "Matching of affinely invariant regions for visual servoing," in *Proc. Robotics and Automation, IEEE International Conference on*, vol. 2, pp. 1601–1606, 1999.

[57] L. Van Gool, T. Moons, and D. Ungureanu, "Affine / photometric invariants for planar intensity patterns," in *Proc. Computer Vision, European Conference on*, Lecture Notes in Computer Science, pp. 642–651, Springer Berlin / Heidelberg, 1996.

[58] L. Van Gool, T. Tuytelaars, and A. Turina, "Local features for image retrieval," in *State-Of-The-Art in Content-Based Image and Video Retrieval*

(R. C. Veltkamp, H. Burkhardt, and H.-P. Kriegel, eds.), pp. 21–41, Kluwer Academic Publishers, 2001.

[59] I. Young and L. van Vliet, "Recursive implementation of the gaussian filter," *Signal Processing*, vol. 44, no. 2, pp. 139–151, 1995.

Appendix A

Algorithm Derivations

This appendix lists the derivation of several components of the WiDense system presented in Chapter 7, Section 7.1.

A.1 Decomposition of an Affine Transformation

Local image features are defined by affine normalisation transformations. The generic form of this transformation is (from Chapter 3, Section 3.2.2),

$$\mathbf{H}\left(t_x, t_y, q, \phi, \theta, k\right) = \mathbf{T}\left(t_x, t_y\right)\mathbf{A}\left(q, \phi\right)\mathbf{R}\left(\theta\right)\mathbf{K}\left(k\right). \tag{A.1}$$

The inverse compositional image alignment algorithms (presented later in this section) do not maintain this parameterisation of the feature transformation and only returns the transformation matrix. Several parts of the WiDense algorithm require the matrix to be decomposed into the above parameterisation. An affine transformation matrix with unknown parameters may be decomposed into the above form using the procedure described below.

The centre coordinates, t_x and t_y, can be extracted from the last column of $\mathbf{H}$ directly. The scale component can be extracted by computing the square root of the determinant of the upper left 2×2 block of $\mathbf{H}$, or, $k = \sqrt{\det\left(\mathbf{T}^{-1}\mathbf{H}\right)}$. The remaining rotation and shape components may be separated as follows,

$$\begin{aligned}
\mathbf{H}_{AR} &= \mathbf{K}^{-1}\mathbf{T}^{-1}\mathbf{H} = \mathbf{A}\mathbf{R}, \\
\mathbf{H}_{AR}\mathbf{H}_{AR}^{\top} &= \mathbf{A}\mathbf{R}\mathbf{R}^{\top}\mathbf{A}^{\top} = \mathbf{A}^{2} \\
&= \begin{bmatrix} \mathbf{A}'^{2} & \mathbf{0} \\ \mathbf{0}^{\top} & 1 \end{bmatrix},
\end{aligned}$$

where $\mathbf{A}'^{2}$ a 2×2 matrix. An eigen decomposition of $\mathbf{A}'^{2}$ yields q^2 as the smallest eigenvalue and ϕ as the angle of the eigenvector associated with q^2. The rotation matrix may be recovered by computing $\mathbf{A}'^{-1}$ from q^{-1} and ϕ, and computing,

$$\mathbf{R} = \begin{bmatrix} \mathbf{A}'^{-1} & \mathbf{0} \\ \mathbf{0}^{\top} & 1 \end{bmatrix} \mathbf{H}_{AR}.$$

The rotation angle, θ, may then be recovered by computing the arccos of the top left element of $\mathbf{R}$ and taking the sign of the middle left element.

A.2 Inverse Compositional Image Alignment using an Affine Geometric Transformation and a Linear Intensity Transformation

This section presents the derivation of the inverse compositional image alignment algorithm using an affine transformation, followed by the derivation of the inverse compositional image intensity alignment algorithm. A combined algorithm that performs geometric and photometric alignment simultaneously is presented after the derivations.

The objective of the inverse compositional image alignment algorithm is to iteratively minimise the cost function,

$$e_w = \sum_{\forall \mathbf{x}} \left(I_t \left(\mathbf{W} \left(\mathbf{x}, \mathbf{p}_\Delta \right) \right) - I_r \left(\mathbf{W} \left(\mathbf{x}, \mathbf{p} \right) \right) \right)^2 ,$$

with respect to $\mathbf{p}_\Delta$. Here I_t is referred to as the template image, I_r is the registration image, $\mathbf{W}(\mathbf{x}, \mathbf{p})$ is some transformation of the coordinates, $\mathbf{x}$, with parameters $\mathbf{p}$, and $\mathbf{p}_\Delta$ is referred to as the update parameters.

Using an affine coordinate transformation,

$$\mathbf{H}(\mathbf{x}, \mathbf{p}) = \mathbf{H}(\mathbf{p})\,\mathbf{x} = \begin{bmatrix} 1+p_1 & p_2 & p_3 \\ p_4 & 1+p_5 & p_6 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x},$$

the cost function can be written as,

$$e_w = \sum_{\forall \mathbf{x}} \left(I_t \left(\mathbf{H}(\mathbf{p}_\Delta)\,\mathbf{x} \right) - I_r \left(\mathbf{H}(\mathbf{p})\,\mathbf{x} \right) \right)^2 .$$

The first order Taylor expansion of this equation is,

$$e_w = \sum_{\forall \mathbf{x}} \left(I_t \left(\mathbf{H}(\mathbf{0})\,\mathbf{x} \right) + \mathbf{D}\mathbf{p}_\Delta - I_r \left(\mathbf{H}(\mathbf{p})\,\mathbf{x} \right) \right)^2 , \tag{A.2}$$

where,

$$\begin{aligned} \mathbf{D} &= \nabla I_t \left(\mathbf{H}(\mathbf{0})\,\mathbf{x} \right) \tfrac{\partial \mathbf{H}(\mathbf{p})\mathbf{x}}{\partial \mathbf{p}} \\ &= \left[\begin{array}{cccccc} \tfrac{\partial I_t}{\partial x} x & \tfrac{\partial I_t}{\partial x} y & \tfrac{\partial I_t}{\partial x} & \tfrac{\partial I_t}{\partial y} x & \tfrac{\partial I_t}{\partial y} y & \tfrac{\partial I_t}{\partial y} \end{array} \right], \end{aligned} \tag{A.3}$$

and,

$$\begin{aligned} \mathbf{H}(\mathbf{0}) &= \mathbf{I}, \\ \nabla I_t(\mathbf{x}) &= \left[\begin{array}{cc} \tfrac{\partial}{\partial x} & \tfrac{\partial}{\partial y} \end{array} \right] I_t(\mathbf{x}), \\ \tfrac{\partial \mathbf{H}(\mathbf{p})\mathbf{x}}{\partial \mathbf{p}} &= \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}. \end{aligned}$$

The partial derivative of Equation A.2 with respect to $\mathbf{p}_\Delta$ is,

$$\frac{\partial e_w}{\partial \mathbf{p}_\Delta} = \sum_{\forall \mathbf{x}} \mathbf{D}^\top \left(I_t(\mathbf{x}) + \mathbf{D}\mathbf{p}_\Delta - I_r \left(\mathbf{H}(\mathbf{p})\,\mathbf{x} \right) \right).$$

The parameter updates, $\mathbf{p}_\Delta$, may be computed by setting this equation to zero and solving for $\mathbf{p}_\Delta$,

$$\mathbf{p}_\Delta = \mathcal{H}_g^{-1} \sum_{\mathbf{x}} \mathbf{D}^\top I_e, \tag{A.4}$$

where,

$$I_e = I_r\left(\mathbf{H}\left(\mathbf{p}\right)\mathbf{x}\right) - I_t\left(\mathbf{x}\right),$$

$$\mathcal{H}_g = \sum_{\mathbf{x}} \mathbf{D}^\top \mathbf{D}.$$

Once the update parameters have been computed, the transformation is updated according to,

$$\mathbf{H}\left(\mathbf{p}\right) \leftarrow \mathbf{H}\left(\mathbf{p}\right)\mathbf{H}^{-1}\left(\mathbf{p}_\Delta\right).$$

It is assumed that the geometric alignment between the images is not grossly inaccurate when the photometric image alignment is performed. A linear function of image intensities is used to align the images. At each iteration of the alignment algorithm, the error function,

$$e_i = \sum_{\forall \mathbf{x}} \left(i_{\Delta a} I_t\left(\mathbf{x}\right) + i_{\Delta b} - i_a I_r\left(\mathbf{x}\right) - i_b\right)^2,$$

is minimised with respect to $\mathbf{i}_\Delta = \left[\begin{array}{cc} i_{\Delta a} & i_{\Delta b} \end{array}\right]$. The partial derivatives of this function are,

$$\frac{\partial e_i}{\partial \mathbf{i}_\Delta} = \sum_{\forall \mathbf{x}} \left[\begin{array}{c} I_t\left(\mathbf{x}\right) \\ 1 \end{array}\right]\left(i_{\Delta a} I_t\left(\mathbf{x}\right) + i_{\Delta b} - i_a I_r\left(\mathbf{x}\right) - i_b\right).$$

The parameter updates, $\mathbf{i}_\Delta$, may be computed by setting this equation to zero and solving for $\mathbf{i}_\Delta$,

$$\mathbf{i}_\Delta = \mathcal{H}_i^{-1} \sum_{\forall \mathbf{x}} \left[\begin{array}{c} I_t\left(\mathbf{x}\right) I_r'\left(\mathbf{x}\right) I_r'\left(\mathbf{x}\right) \end{array}\right], \tag{A.5}$$

where,

$$\mathcal{H}_i = \sum_{\forall \mathbf{x}} \left[\begin{array}{cc} I_t^2\left(\mathbf{x}\right) & I_t\left(\mathbf{x}\right) \\ I_t\left(\mathbf{x}\right) & 1 \end{array}\right],$$

$$I_r^{'}(\mathbf{x}) = i_a I_r(\mathbf{x}) + i_b.$$

The parameters are then updated according to,

$$\begin{aligned} I_r^{'}(\mathbf{x}) &\leftarrow i_{\Delta a} I_r^{'}(\mathbf{x}) + i_{\Delta b}, \\ \therefore i_a &\leftarrow i_a \cdot i_{\Delta a}, \\ i_b &\leftarrow i_b \cdot i_{\Delta a} + i_{\Delta b}. \end{aligned}$$

Algorithm A.1 lists the combined affine geometric and linear photometric image alignment algorithm. Notice that the photometric transformation is only updated if the update parameter $i_{\Delta a}$ is in the range $i_{\Delta a} \in (0.4, 2.5)$ (line A.1.12). This is to prevent extreme intensity changes that may result from attempting photometric alignment when the geometric alignment is not yet sufficiently accurate.

Algorithm A.1: Iterative Inverse Compositional Image Alignment using an Affine Geometric Transformation and a Linear Intensity Transformation.

```
Function:
$(\mathbf{H}_r, \mathbf{i}_r, e_o) \leftarrow \text{ALIGNAP}\left(I_t(\mathbf{x}), I_r(\mathbf{x}), \mathbf{H}_r, \mathbf{i}_r, n_i, t_\Delta\right)$.
Description:
Iteratively refines $\mathbf{H}_r$ and $\mathbf{i}_r$ to minimise the sum of squared
differences between $I_t(\mathbf{x})$ and $i_a I_r(\mathbf{H}_r\mathbf{x}) + i_b$.
Input:
$I_t(\mathbf{x})$ – The template image.
$I_r(\mathbf{x})$ – The registration image.
$\mathbf{H}_r$ – The affine registration transformation.
$\mathbf{i}_r = \left[\begin{array}{cc} i_a & i_b \end{array}\right]^\top$ – The linear photometric transformation.
$n_i$ – The maximum allowed number of iterations.
$t_\Delta$ – The convergence threshold.
Output:
$\mathbf{H}_r$ – the refined affine registration transformation.
$\mathbf{i}_r$ – the refined linear photometric transformation.
$e_o$ – The final RMS difference between $I_t$ and transformed $I_r$.
A.1.1  begin
A.1.2      compute $\mathbf{D}$ from Equation A.3.
A.1.3      compute $\mathcal{H}_g^{-1}$ from Equation A.4.
A.1.4      compute $\mathcal{H}_i^{-1}$ from Equation A.5.
A.1.5      $j \leftarrow 0$.
A.1.6      repeat
A.1.7          $I_r'(\mathbf{x}) \leftarrow i_a I_r(\mathbf{H}_r\mathbf{x}) + i_b$.
A.1.8          $I_e(\mathbf{x}) = I_r' - I_t$.
A.1.9          $\mathbf{p}_\Delta \leftarrow \mathcal{H}_g^{-1} \sum_{\mathbf{x}} \mathbf{D}^\top (I_e(\mathbf{x}))$.
A.1.10         $\mathbf{H}_r \leftarrow \mathbf{H}_r \mathbf{H}^{-1}(\mathbf{p}_\Delta)$.
A.1.11         $\mathbf{i}_\Delta \leftarrow \mathcal{H}_i^{-1} \sum_{\forall\mathbf{x}} \left[\begin{array}{c} I_t(\mathbf{x}) I_r'(\mathbf{x}) \\ I_r'(\mathbf{x}) \end{array}\right]$.
A.1.12         if $(i_{\Delta a} > 0.4) \cdot (i_{\Delta a} < 2.5)$ then
A.1.13             $i_a \leftarrow i_a i_{\Delta a}$.
A.1.14             $i_b \leftarrow i_b i_{\Delta a} + i_{\Delta b}$.
A.1.15         end
A.1.16         $p_\Delta \leftarrow (i_{\Delta a} - 1)^2 + i_{\Delta b}^2 + \mathbf{p}_\Delta^\top \mathbf{p}_\Delta$.
A.1.17         $j \leftarrow j + 1$.
A.1.18     until $(j > n_i) + (p_\Delta < t_\Delta)$
A.1.19     $e_o \leftarrow \sqrt{N_{\mathbf{x}}^{-1} \sum_{\forall\mathbf{x}} I_e^2(\mathbf{x})}$.
A.1.20 end
```

A.3 Inverse Compositional Image Alignment using a Translation Transformation

The inverse compositional image alignment algorithm using a translation transformation is derived in a similar manner as the affine and photometric alignment algorithm in the preceding section, but is considerably less complex. The translation registration transformation is parameterised as,

$$\mathbf{T}(\mathbf{p}) = \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \\ 0 & 0 & 1 \end{bmatrix}.$$

The first order Taylor expansion of the objective function is,

$$e_w = \sum_{\forall \mathbf{x}} \left(I_t(\mathbf{x}) + \mathbf{D}\mathbf{p}_\Delta - I_r(\mathbf{T}(\mathbf{p})\mathbf{x}) \right)^2,$$

where,

$$\mathbf{D} = \nabla I_t(\mathbf{x}) \frac{\partial \mathbf{T}(\mathbf{p})\mathbf{x}}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial I_t}{\partial x} & \frac{\partial I_t}{\partial y} \end{bmatrix}, \tag{A.6}$$

and,

$$\begin{aligned} \nabla I_t(\mathbf{x}) &= \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} I_t(\mathbf{x}), \\ \frac{\partial \mathbf{T}(\mathbf{p})\mathbf{x}}{\partial \mathbf{p}} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

The parameter updates, $\mathbf{p}_\Delta$, may be computed by setting the first order partial derivatives of this equation to zero and solving for $\mathbf{p}_\Delta$,

$$\mathbf{p}_\Delta = \mathcal{H}_g^{-1} \sum_{\forall \mathbf{x}} \mathbf{D}^\top I_e, \tag{A.7}$$

where,

$$I_e = I_r(\mathbf{T}(\mathbf{p})\mathbf{x}) - I_t(\mathbf{x}),$$

$$\mathcal{H}_g = \sum_{\forall \mathbf{x}} \mathbf{D}^\top \mathbf{D}.$$

Algorithm A.2: Iterative Inverse Compositional Image Alignment using a Translation Transformation.

Function:
$(\mathbf{T}_r, e_o) \leftarrow \text{ALIGNT}\left(I_t(\mathbf{x}), I_r(\mathbf{x}), \mathbf{T}_r, n_i, t_\Delta\right)$.
Description:
Iteratively refines $\mathbf{T}_r$ to minimise the sum of squared differences between $I_t(\mathbf{x})$ and $I_r(\mathbf{T}_r\mathbf{x})$.
Input:
$I_t(\mathbf{x})$ – The template image.
$I_r(\mathbf{x})$ – The registration image.
$\mathbf{T}_r$ – The translation transformation.
n_i – The maximum allowed number of iterations.
t_Δ – The convergence threshold.
Output:
$\mathbf{T}_r$ – the refined translation transformation.
e_o – The final RMS difference between I_t and transformed I_r.

```
A.2.1  begin
A.2.2      compute $\mathbf{D}$ from Equation A.6.
A.2.3      compute $\mathcal{H}_g^{-1}$ from Equation A.7.
A.2.4      $j \leftarrow 0$.
A.2.5      repeat
A.2.6          $I_r'(\mathbf{x}) \leftarrow I_r(\mathbf{T}_r\mathbf{x})$.
A.2.7          $I_e(\mathbf{x}) = I_r' - I_t$.
A.2.8          $\mathbf{p}_\Delta \leftarrow \mathcal{H}_g^{-1} \sum_{\mathbf{x}} \mathbf{D}^\top (I_e(\mathbf{x}))$.
A.2.9          $\mathbf{T}_r \leftarrow \mathbf{T}_r \mathbf{T}^{-1}(\mathbf{p}_\Delta)$.
A.2.10         $p_\Delta \leftarrow \mathbf{p}_\Delta^\top \mathbf{p}_\Delta$.
A.2.11         $j \leftarrow j + 1$.
A.2.12     until $(j > n_i) + (p_\Delta < t_\Delta)$
A.2.13     $e_o \leftarrow \sqrt{N_{\mathbf{x}}^{-1} \sum_{\forall \mathbf{x}} I_e^2(\mathbf{x})}$.
A.2.14 end
```

Once the update parameters have been computed, the transformation is updated according to,

$$\mathbf{T}(\mathbf{p}) \leftarrow \mathbf{T}(\mathbf{p})\,\mathbf{T}^{-1}(\mathbf{p}_\Delta).$$

The inverse compositional image alignment algorithm using a translation transformation is listed in Algorithm A.2.

A.4 Criteria for Subdividing Eccentric Features

Step 7 of the algorithm presented in Section 7.1.2 lists three criteria that must be met before a corresponding pair of features is subdivided. These are derived here.

Criterion 1: $k >= 24$ pixels. Features must be large enough to be aligned effectively using inverse compositional alignment. The threshold of 24 pixels is chosen so that regions are at least as large as the chosen width of the template image used in the alignment process ($2r_t + 3$, with $r_t = 20$).

Criterion 2: $q_1q_2 < 0.69$. If $q_1q_2 > 2^{\frac{-1}{2}}$, then the subdivided features will be more eccentric than the original features. If both q_1 and q_2 are close to $2^{\frac{-1}{4}}$, then a very large number of subdivisions may be required before $q_1q_2 > 2^{\frac{-1}{2}}$. For this reason the threshold is set slightly lower, to 0.69.

Criterion 3: $q_{12} < q_d$, where q_{12} is the eccentricity parameter of matrix $\mathbf{H}_{12} = \mathbf{H}_1^{-1}\mathbf{H}_2$ and q_d is the eccentricity parameter of matrix,

$$\mathbf{A}_d = \mathbf{A}\left(q_1, \phi_1 + \theta_1\right)\mathbf{A}\left(q_2, \phi_2 + \theta_2\right). \tag{A.8}$$

The third criterion requires that the change in viewpoint not account for all of the eccentricity of the normalisation transformations. This is determined as follows. The direct transformation between a pair of corresponding features is computed from their normalisation transformations, $\mathbf{H}_1$ and $\mathbf{H}_2$, as, $\mathbf{H}_{12} = \mathbf{H}_1^{-1}\mathbf{H}_2$, so that $\mathbf{H}_{12}$ maps feature 1 to feature 2. The total image deformation in the feature transformations is computed by Equation A.8. The eccentricity of matrix $\mathbf{A}_d$ will be larger than the eccentricity of $\mathbf{H}_{12}$ if the features are more eccentric than the transformation mapping one feature to the other. The eccentricity parameters, q_{12} and q_d may be extracted from the matrices $\mathbf{H}_{12}$ and $\mathbf{A}_d$ using the method detailed in Section A.1.

A.5 Computing the Parameters of Subdivided Features

Section 7.1.2 presents a method for subdividing eccentric features into less eccentric features (step 7 in the algorithm). This section presents the derivation of the formulae for computing the parameters of the sub-features $\mathbf{H}_{a1}$, $\mathbf{H}_{a2}$, $\mathbf{H}_{b1}$ and $\mathbf{H}_{b2}$, given two corresponding features, $\mathbf{H}_a$ and $\mathbf{H}_b$, with $q_a < q_b$.

The objective of feature splitting is to compute two features that have the same combined support region as the original feature. The orientation (parameter θ) of the new features is unimportant, as long as it is consistent across the correspondence, hence, $\theta_{a1} = \theta_{a2} = \theta_a$.

The principal axes of the new features should be parallel with the original features' axes, meaning ϕ is preserved, $\phi_{u1} = \phi_{u2} = \phi_a$.

The area of the feature support region is proportional to k^2. Therefore, if the feature is split in half, then the new scale parameter is found as follows,

$$\begin{aligned} 2k_{a1}^2 = 2k_{a2}^2 &= k_a^2, \\ k_{a1} = k_{a2} &= \tfrac{1}{\sqrt{2}} k_a. \end{aligned}$$

The length of the longest axis of the feature is divided in two, so that,

$$\begin{aligned} 2k_{a1} q_{a1}^{-1} &= k_a q_a^{-1}, \\ \tfrac{2}{\sqrt{2}} k_a q_{a1}^{-1} &= k_a q_a^{-1}, \\ q_{a1}^{-1} &= \tfrac{1}{\sqrt{2}} q_a^{-1}, \\ q_{a1} = q_{a2} &= \sqrt{2} q_a. \end{aligned}$$

Let the centre points of features $a1$ and $a2$ be defined as, $\mathbf{t}_{a1} = \begin{bmatrix} t_{xa1} & t_{ya1} & 1 \end{bmatrix}^\top$,

$\mathbf{t}_{a2} = \begin{bmatrix} t_{xa2} & t_{ya2} & 1 \end{bmatrix}^\top$. Define two points $\mathbf{t}'_{a1}$ and $\mathbf{t}'_{a2}$ that are related to $\mathbf{t}_{a1}$ and $\mathbf{t}_{a2}$ by, $\mathbf{t}_{a1} = \mathbf{H}_a\mathbf{t}'_{a1}$, $\mathbf{t}_{a2} = \mathbf{H}_a\mathbf{t}'_{a2}$.

The points $\mathbf{t}'_{a1}$ and $\mathbf{t}'_{a2}$ are located at a distance of 0.5 (in normalised coordinates) from the coordinate origin. $\mathbf{H}_a$ may be expressed as,

$$\mathbf{H}_a = \mathbf{T}\left(t_{xa}, t_{ya}\right)\mathbf{K}\left(k_a\right)\mathbf{R}\left(-\phi_a\right)\mathbf{D}\left(q_a\right)\mathbf{R}\left(\phi_a\right)\mathbf{R}\left(\theta_a\right),$$

which is equivalent to the feature transformation model presented in Equation 3.1, Chapter 3, Section 3.2.2. Applying the rotations $\mathbf{R}\left(\phi_a\right)\mathbf{R}\left(\theta_a\right)$ to $\mathbf{t}'_{a1}$ and $\mathbf{t}'_{a2}$ will place these points on the y-axis, such that, $\mathbf{R}\left(\phi_a\right)\mathbf{R}\left(\theta_a\right)\mathbf{t}'_{a1} = \begin{bmatrix} 0 & 0.5 & 1 \end{bmatrix}^\top$, and $\mathbf{R}\left(\phi_a\right)\mathbf{R}\left(\theta_a\right)\mathbf{t}'_{a2} = \begin{bmatrix} 0 & -0.5 & 1 \end{bmatrix}^\top$. The location of the new centre points can be found by applying the remainder of the transformation,

$$\begin{aligned} \mathbf{t}_{a1} &= \mathbf{T}\left(t_{xa}, t_{ya}\right)\mathbf{K}\left(k_a\right)\mathbf{R}\left(-\phi_a\right)\mathbf{D}\left(q_a\right)\begin{bmatrix} 0 \\ 0.5 \\ 1 \end{bmatrix}, \\ &= \begin{bmatrix} t_{xa} \\ t_{ya} \\ 1 \end{bmatrix} + 0.5k_a q_a^{-1}\begin{bmatrix} -\sin\left(\phi_a\right) \\ \cos\left(\phi_a\right) \\ 1 \end{bmatrix}. \end{aligned}$$

Similarly,

$$\mathbf{t}_{a2} = \begin{bmatrix} t_{xa} \\ t_{ya} \\ 1 \end{bmatrix} - 0.5k_a q_a^{-1}\begin{bmatrix} -\sin\left(\phi_a\right) \\ \cos\left(\phi_a\right) \\ 1 \end{bmatrix}.$$

The equations in Section 7.1.2, step 7 express the above equations in a manner more convenient for implementation.

The features $\mathbf{H}_{b1}$ and $\mathbf{H}_{b2}$ corresponding to $\mathbf{H}_{a1}$ and $\mathbf{H}_{a2}$ can be found by composing $\mathbf{H}_{a1}$ and $\mathbf{H}_{a2}$ from the parameters found above, and then projecting them to image b. The direct transformation from feature a to b is, $\mathbf{H}_{ab} = \mathbf{H}_a^{-1}\mathbf{H}_b$. The corresponding features are then,

$$\begin{aligned}\mathbf{H}_{b1} &= \mathbf{H}_{ab}\mathbf{H}_{a1},\\ \mathbf{H}_{b2} &= \mathbf{H}_{ab}\mathbf{H}_{a2}.\end{aligned}$$

Finally, the intensity mapping parameters are simply copied,

$$\begin{aligned}i_{ua1} = i_{ua2} &= i_{ua},\\ i_{va1} = i_{va2} &= i_{va}.\end{aligned}$$

A.6 Computing the Parameters of Replicated Features

Section 7.1.3 presents a procedure for aligning neighbouring regions of a feature by replicating the feature in all directions. This section lists the derivation of the equations used to compute the parameters of a replicated feature.

The objective is to replicate a corresponding pair of features, $\mathbf{H}_1$ and $\mathbf{H}_2$, in direction $\mathbf{d} = \begin{bmatrix} d_x & d_y & 1 \end{bmatrix}^\top$, where $d_x, d_y \in \{-1, 0, 1\}$, to produce a pair of features, $\mathbf{H}_{r1}$ and $\mathbf{H}_{r2}$. The only parameters in the replicated features that are different to the original features are the position (translation) parameters, t_x and t_y. These are found using a method similar to computing the position of subdivided features, as in the previous section. The only differences are the direction and distance of displacement. The direction is determined by $\mathbf{d}$. The distance is chosen such that the new features are next to the original features, with a small amount of overlap. When processing image patches, the margin of the image cannot be used. Including limited overlap between features ensures that all of the image is processed in subsequent steps.

Define the centre point of feature r_1 as the point, $\mathbf{t}_{r1} = \begin{bmatrix} t_{xr1} & t_{yr1} & 1 \end{bmatrix}^\top$, and define a point $\mathbf{t}_{r1}^{'}$ that is related to $\mathbf{t}_{r1}$ by $\mathbf{t}_{r1} = \mathbf{H}_1\mathbf{t}_{r1}^{'}$. Let the point $\mathbf{t}_{r1}^{'}$ satisfy the equation,

$$\mathbf{R}\left(\phi_1 + \theta_1\right)\mathbf{t}_{r1}^{'} = \begin{bmatrix} 1.6d_x \\ 1.6d_y \\ 1 \end{bmatrix}.$$

The constant, 1.6, in this equation determines the relative distance of the original and replicated features. A value of 2 would result in these features being adjacent, whereas 1.6 results in some overlap.

The location of the new centre points can be found by applying the remainder of the transformation,

$$\mathbf{t}_{r1} = \mathbf{T}\left(t_{x1}, t_{y1}\right)\mathbf{K}\left(k_1\right)\mathbf{R}\left(-\phi_1\right)\mathbf{D}\left(q_1\right)\begin{bmatrix} 1.6d_x \\ 1.6d_y \\ 1 \end{bmatrix}.$$

Expanding the above gives,

$$\begin{aligned} t_{xp1} &= t_{x1} + 1.6k_1\left(d_x q_1 \cos\left(\phi_1\right) - d_y q_1^{-1} \sin\left(\phi_1\right)\right), \\ t_{yp1} &= t_{ya} + 1.6k_1\left(d_x q_1 \sin\left(\phi_1\right) + d_y q_1^{-1} \cos\left(\phi_1\right)\right). \end{aligned}$$

The corresponding feature, $\mathbf{H}_{r2}$ may be computed from the relative transformation as,

$$\begin{aligned} \mathbf{H}_{12} &= \mathbf{H}_1^{-1}\mathbf{H}_2, \\ \mathbf{H}_{r2} &= \mathbf{H}_{12}\mathbf{H}_{r1}. \end{aligned}$$