Content-based Analysis of Video Through Sparse Features

A dissertation submitted to the University of Dublin
for the degree of Doctor of Philosophy

Dan Ring
Trinity College Dublin, March 2010

Signal Processing and Media Applications
Department of Electronic and Electrical Engineering
Trinity College Dublin

Declaration

I hereby declare that this thesis has not been submitted as an exercise for a degree at this or any other University and that it is entirely my own work.

I agree that the Library may lend or copy this thesis upon request.

Signed,

Dan Ring

March 24, 2010.

Abstract

Recent advances in technology have significantly advanced the ability of the average user to quickly acquire, produce, and disseminate large amounts of video data. The problem now is how to automatically extract useful information, allowing easier access and manipulation of this data. This thesis investigates the use of local image features in representing and understanding image content in video for novel approaches to traditional applications in video.

The first feature enhanced application presented is a simple, computationally efficient method based on implicit motion analysis for accurately detecting repetitive events in sports coaching video. The aim is to allow rapid record and review of characteristic actions, such as tennis serves, golf swings, cricket bats etc., in order to hone athletic technique. However, it can be difficult for the system to distinguish between interesting sports actions and unintended player movements. Local image features are exploited to represent and compare large amounts of image content from the parsed events. Using information retrieval techniques to automatically detect false alarms, the overall detection accuracy is significantly improved.

Part of this thesis addresses one of the main practical considerations of using local features; how to select the most appropriate feature detection system for a given application. There are many impressive comparative surveys on the performance of various feature detectors, however different feature detectors behave differently depending on the source image data. This thesis presents a novel protocol to allow the user to easily and systematically compare wide ranges of feature detectors and parameters, using the actual source imagery of the intended algorithm as the only input to the system. This application specific approach gives a more realistic measure of actual feature detector performance.

Lastly, local image features are applied to the difficult task of semi-automatic object cut out for video post production. Extracting an object from video is a time consuming and tedious task with few tools to help automate the workflow of the user. Recent impressive examples of video object segmentation in research use colour as the primary likelihood for what is considered foreground or background. However, for many reasons, colour is not a particularly stable feature space over time, requiring frequent corrections to achieve a quality cut out. This thesis presents a novel method to propagate user supplied information more efficiently throughout the video, dramatically reducing the amount of time and manual effort required to accurately segment an object in video.

Acknowledgments

Over the last few years of post-graduate life in Sigmedia, I've had the great pleasure of working with many interesting and talented people, who deserve a great deal more thanks than a single acknowledgements page can provide. First and foremost, I would like to thank my supervisor, Prof. Anil Kokaram, for all the guidance, support and advice throughout the past four years.

A huge thank you must also be given to those who have been endlessly called on for assistance, in particular Dr. François Pitié, Dr. David Corrigan, Dr. Rozenn Dahyot and Dr. Naomi Harte. A special thank you goes to everyone in the Sigmedia lab that made it both a great place to work and socialise, (in alphabetical order) they are Daire Lennon, Damien Kelly, Dr. Hugh Denman, John Squires and Riccardo Veglianti. In addition, a marked thank you to Gavin Kearney for all the great chats, advice and laughs. Very special thanks go to Deirdre O'Regan for being a fantastic PhD partner-in-crime through many sorties, and a great friend over the last near-decade.

This work could not have been accomplished without the greatly appreciated financial support of the Irish Research Council for Science, Engineering and Technology (IRCSET) under the Embark Initiative. Additionally, part of this work has been funded by a summer internship in Adobe Systems Inc. I would like to extend a huge thank you to everyone I have had the pleasure of working with in Adobe, in particular David Simons, the After Effects group, and the (at the time) DMRG, ATL and GRG groups.

I also whole-heartedly thank my family for always being supportive, understanding, and a great source of happiness. Lastly, this thesis could not have been accomplished without the love and encouragement of my significant other, Kate O'Gorman, who has listened to more about local image features than anyone should have to endure, thank you.

Contents

List of Acronyms **xiii**

1 Introduction **1**

1.1 Contributions of this thesis 5

1.2 Publications 5

2 On Local Image Features **7**

2.1 Introduction to Local Image Features 9

2.1.1 Correspondence Matching 9

2.1.2 Content-based Classification 10

2.2 Modern Approaches to Feature Detection 12

2.2.1 Detecting Local Features 12

2.3 Describing Local Features 19

2.4 Difficulties in Identifying Correct Feature Matches 21

2.4.1 Finding Matches 21

2.4.2 Selecting Detectors & Descriptors 25

2.5 Addressing Issues in Practical Feature-based Systems 26

2.5.1 Incorporating Spatial-Context 27

2.5.2 Improving Nearest-Neighbour Descriptor Matching 30

2.5.3 Database Matching 32

2.5.4 Comparing Feature Detector Performance 36

2.6 Summary 38

3 A Protocol for Feature-Detector Comparison **39**

3.1 Introduction 39

3.2 Feature Comparison Protocol 40

3.2.1 Characterisation of Detector Performance 41

3.2.2 Measuring Detector Performance 43

3.2.3 Experimental Validation 44

3.3 Examples & Results 48

3.3.1 Example Application Results . . . 51
3.4 Discussion & Conclusion . . . 52

4 Motion Cues for On-Line Event Parsing 55
4.1 Previous Work . . . 56
4.2 Analysing Intrinsic Motion . . . 60
4.2.1 Analysing Movement . . . 62
4.3 Analysing the Movement Signal . . . 67
4.4 Identifying Interesting Player Actions . . . 70
4.5 Parameter Selection . . . 73
4.6 Experimental Results . . . 77
4.6.1 Frame-based Numerical Results . . . 78
4.6.2 Action-based Numerical Results . . . 78
4.6.3 Assessment of System Design . . . 82
4.7 Using Local Image Features to Detect False Alarms . . . 86
4.7.1 Visual Results . . . 91
4.8 Discussion . . . 91

5 A Review of Interactive Object Cut Out Techniques 97
5.1 Object Cut Out; A History . . . 99
5.2 Still Image Segmentation . . . 103
5.2.1 Grab-Cut . . . 105
5.2.2 Distance-Cut . . . 107
5.3 Rotoscoping vs. Segmentation in Video . . . 109
5.3.1 Spline based Cut Out . . . 110
5.3.2 Pixel based Cut out . . . 114
5.4 Matting . . . 126
5.4.1 Bayesian Matting . . . 127
5.4.2 Closed-Form Matting . . . 129
5.5 Discussion & Conclusion . . . 132
5.5.1 Quality of Results . . . 132
5.5.2 User Interaction . . . 133
5.5.3 System Failure . . . 134
5.5.4 On Updating . . . 136
5.5.5 Improving Object Cut-out with Local Image Features . . . 136

6 Feature Based Object Segmentation 139
6.1 Introduction . . . 139
6.2 Information Retrieval Assisted User-Interaction . . . 141
6.2.1 IR System Specifics . . . 142

6.2.2 Measuring Object Detection Reliability 143
6.3 Feature-Based Matte Propagation 145
6.3.1 Identifying Feature Correspondences 145
6.3.2 Speeding-Up Nearest Neighbour Descriptor Matching 146
6.3.3 Propagating Mattes 148
6.3.4 Refining the Partial Matte 150
6.3.5 Parameter Selection 152
6.3.6 User Interaction 154
6.4 Results 156
6.4.1 Statistical Evaluation 158
6.4.2 Visual Evaluation 160
6.5 Discussion 161

7 User Assisted Feature Matching 175
7.1 Introduction 175
7.2 User-Assisted Matching Framework 178
7.2.1 Energy Minimisation 180
7.3 Results & Discussion 180

8 Conclusion 185
8.1 Future Work 186
8.2 Final Remarks 186

A Precision and Recall Measures for Video Parsing 189

B Automatically Assigning Actions to Detection Signals 191

C The Two-Sample Kolmogorov-Smirnov Test Statistic 197

D DVD Contents 201
D.1 Motion Cues for Event Parsing 201
D.2 Feature based Object Segmentation 201

List of Acronyms

AFD	Absolute Frame Difference
CBIR	Content-based Image Retrieval
CDF	Cumulative Distribution Function
DFD	Displaced Frame Difference
DOG	Difference Of Gaussians
DT-CWT	Dual-Tree Complex Wavelet Transform
EM	Expectation Maximisation
HAL	Harris-Laplace
HEL	Hessian-Laplace
HMM	Hidden Markov Model
ICM	Iterated Conditional Modes
IR	Information Retrieval
KS	Kolmorogov-Smirnov
LOG	Laplacian Of Gaussians
MPR	Mean Precision and Recall
MSE	Mean Squared Error
MSER	Maximally Stable Extremal Regions
NCC	Normalised Cross Correlation
NN	Nearest Neighbour
PCA	Principal Component Analysis

PDF	Probability Distribution Function
RANSAC	Random Sample Consensus
SAD	Sum of Absolute Difference
SIFT	Scale Invariant Feature Transform
TFIDF	Term Frequency Inverse Document Frequency
VSM	Vector Space Model

Chapter 1

Introduction

Digital devices have reached the point where the lines between consumer, producer and distributor of content have become blurred. For example; digital video cameras are now inexpensive and convenient to use, personal video recorders (PVR) are issued as standard by cable television companies, mobile phones with cameras can upload video to social media websites instantly, and the verb "to Photoshop" is now a household term for image trickery. Ignoring for a moment the problem of how to store the vast quantities of data, the paramount question now is how to access and manipulate the content of the video.

Research in image processing is often concerned with the "semantic gap"; the differences between how man and machine understand images. Often low level image features, such as colour, edge or motion information, can be combined with specific domain knowledge to accurately capture some intended high level semantic. For example, in soccer footage analysis, the green colour indicates the grass pitch, while the white lines bound regions of interest. The use of semantically low level information can work well in many constrained scenarios, but they are still a long way off from understanding the content of an arbitrary scene of video. *Local image features* attempt to address this problem by detecting and summarising the salient regions of images. Examples of salient regions are corners, blobs, or contiguous patches of similar colour intensities. The idea is that the *image content* can be represented by selecting the most interesting low-level parts of an image, such as the most distinctive corners, or the most isotropic blobs etc. An example of features detected on an image and their corresponding image patches is shown in Figure 1.1. Exploitation of low level features allows local image information at "feature points" as they are often known, to provide "middle level" semantic information. Collections of these feature points are usually considered "sparse", as the distribution of feature point locations is not uniform. For example, not every pixel in the image may have an associated feature, as in the case of typical per-pixel ("dense") motion estimation algorithms.

Patches of image content can now be analysed in the same way pixels would. Although a small collection of pixels does not hold much information, a small collection of image patches is enough to describe an object within an image, and better still, to allow comparisons of the

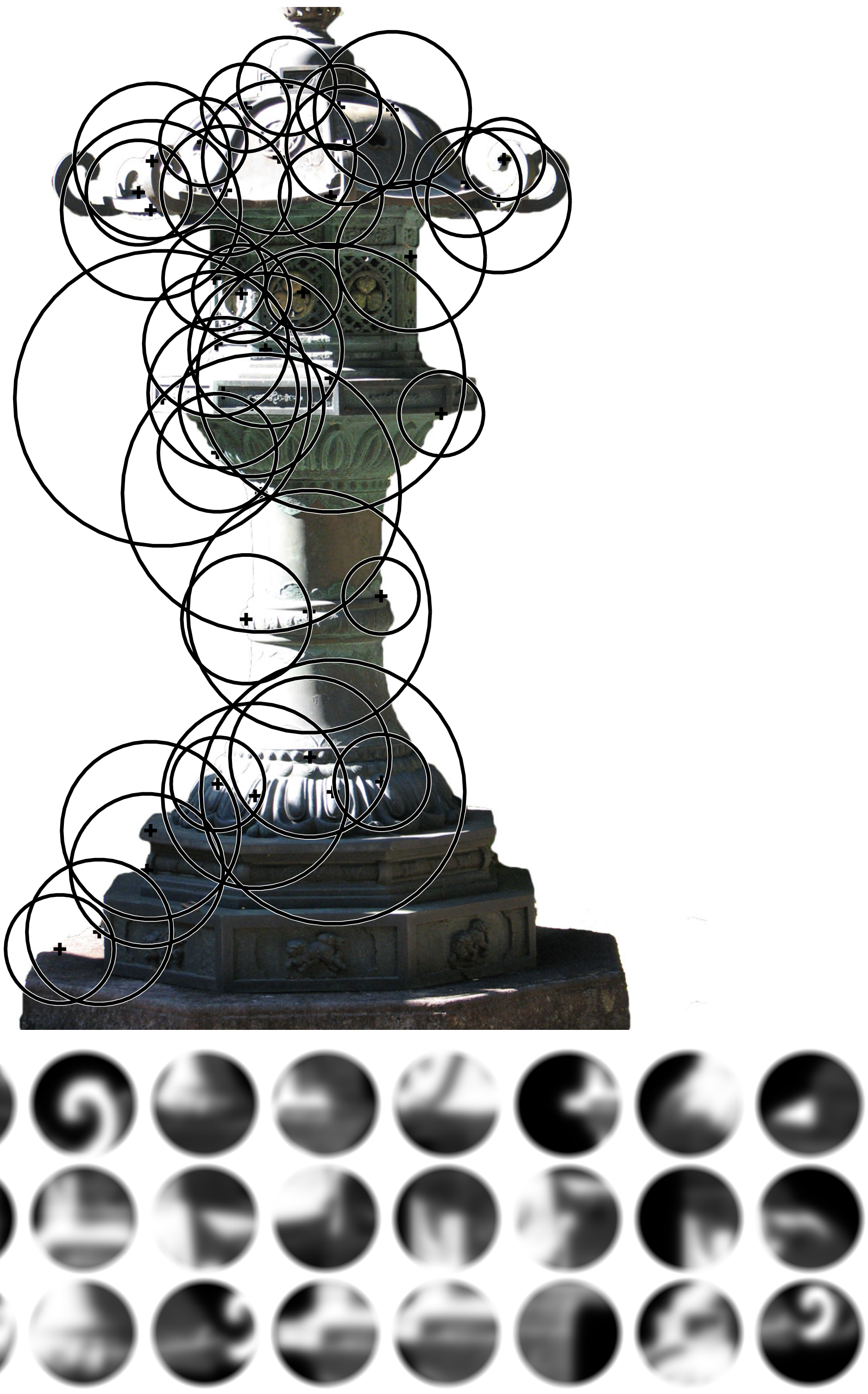


Figure 1.1: Example of using a sparse set of detected salient features (black crosses & circles) to represent the image content. The pixel regions used to describe the features are shown by the collection of circular image patches below. The features in this case are detected using the multi-scale "Harris-Laplace" corner detector.

image content itself. Techniques using these features have been developed to allow comparison of image content at an explicit level, for example identifying correspondences between a pair of images, or at a very general level, such as detecting an arbitrary query object presented by the user throughout a large ($> 1,000,000$ image) database.

The main thrust of this thesis is exploring the use of local image features in applications which in the past have typically used semantically lower level features. The goal is to demonstrate how middle level information from local features can be used in place of, or in conjunction with, other traditional feature spaces to improve performance in common image processing based applications. This thesis is focused on the practical side of local image features in user applications. From this, two secondary objectives arise, and are investigated.

The first is how to select the appropriate feature detection system for a given application. Similar to the choice of edge detectors (i.e. Canny, Sobel, phase congruency etc.), there is no single "feature point detector". The way salient image regions are detected and represented can have a large bearing on application performance. This thesis provides a clear and systematic method for measuring feature detector performance *specific* to the user's application.

The next objective is how to incorporate and exploit user information in feature point based applications, to enable the kind of interactivity usually seen in media post production software. As feature points deal with image content, it is possible to improve the way existing interactive algorithms handle user data. Some applications have an automatic feature based portion of the algorithm to identify similar regions between images with similar content. However, if the feature based portion fails, the rest of the system fails too. Unlike pixel based post-production tasks, the errors cannot simply be corrected by drawing over the erroneous results. This thesis addresses how the user can sensibly and interactively impart information to the system to allow more accurate comparison of the image content.

Chapter 2: On Local Image Features

This chapter introduces the reader to local image features by providing a brief history of the field, followed by some examples of popular, contemporary feature detectors. To put the area into context and to highlight the usefulness of local features, some of the more impressive examples of feature point based applications are then presented. Some of the important issues with using feature points are shown, clarifying and detailing the objectives of this thesis.

Chapter 3: A Protocol for Application Specific Feature-Detector Comparison

This chapter presents a novel protocol for measuring the performance of local feature detection systems. The approach of this system is interesting in that it does not attempt to find "the best" detector, but rather the detector most suited for a *specific* purpose, and is accompanied by experimental justification against ground truth. A variety of popular feature detectors are then evaluated on a selection of common local feature based tasks to effectively illustrate how the protocol can be used.

Chapter 4: Motion Cues for On-Line Event Parsing

This chapter presents a unique, on-line record-and-review application, enabling sports coaches to assess and guide an athlete during a coaching session. The application uses implicit motion cues to automatically parse repetitive sports actions of a single on-screen athlete from a fixed camera without any prior knowledge of the scene or sport being played. The motion features of the system capture the majority of the correct actions, however the false alarm rate is high. The detection accuracy is improved by a simple feature based content comparison technique from Chapter 2, significantly reducing false alarms and improving the overall detection accuracy.

Chapter 5: A Review of Interactive Object Cut Out Techniques

This chapter presents a review of semi-automatic object segmentation systems for post production, emphasising the quality of the object cut out, and how user interaction is incorporated into the system, using many examples from commercial software. A brief historical context is first presented, followed by a discussion of the state of the art in still image and video object cut out systems, with a short introduction to “matting” presented to conclude the typical post production workflow. The merits and deficiencies of the various systems are then summarised, providing the motivation for using local features to improve object segmentation.

Chapter 6: Feature Based Object Segmentation

This chapter proposes a system for performing accurate object segmentation in video through local image features. The ability of local features to be matched across temporally disparate frames in video allows user information to be propagated more effectively throughout a sequence, reducing the overall amount of manual effort required to obtain an object cut out of high quality. Additionally, techniques usually found in feature based object detection, as described in Chapter 2, are repurposed to provide the user with diagnostic information about the segmentation.

Chapter 7: User Assisted Feature Matching

The problem of how the user can interact with local image features is addressed in this short chapter. A semi-automatic method is presented to help encourage correspondences between image regions where feature detection and matching systems typically have difficulty. The proposed system is then evaluated against ground truth images, followed by results in real-world situations.

Chapter 8: Conclusions

This final chapter evaluates the contributions of this thesis, and outlines possible directions for future work.

1.1 Contributions of this thesis

The novel work presented in this thesis can be summarised by the following list:

- A data-centric protocol for the application specific measurement of feature detection system performance, Chapter 3.
- An algorithm for the automatic parsing of sports coaching video based on simple motion cues in the absence of prior information, Chapter 4.
- A simple local feature based shot-clustering algorithm for the automatic detection of dissimilar shots, Chapter 4.
- A local feature based approach for accurately segmenting objects throughout video, Chapter 6.
- A diagnostic method for providing feedback on the effectiveness of the information supplied by the user, using feature based content analysis, Chapter 6.
- An semi-automatic approach for allowing the user to improve feature correspondences between difficult-to-match images, Chapter 7.

1.2 Publications

Portions of the work described in this thesis have appeared in the following publications, ranked first by relevance, then by date. The first two publications in the list (CVMP '09 and ICCV '09) are the most relevant, and have been developed further to form chapters in this thesis. The third publication (CVMP '08) presents the first attempt at using feature points to perform semi-automatic segmentation, which was later developed into the ICCV '09 paper. The fourth and sixth publications (ICIP '07 and SPIE '06) show preliminary work on using intrinsic motion features to detect interesting events. Although not strictly related, the IMVIP '09 publication presents a method for estimating motion between images using feature detection, matching and propagation techniques discussed in this thesis.

- "User-Assisted Feature Correspondence Matching" by Dan Ring and Anil Kokaram, in *IEEE European Conference on Visual Media Production (CVMP)*, London, UK, November 2009.

- "Feature-Cut: Video Object Segmentation Through Local Feature Correspondences" by Dan Ring and Anil Kokaram, in *Proceedings of International Conference of Computer Vision (ICCV) Workshop on Video-oriented Object and Event Classification (VOEC)*, Kyoto, Japan, October 2009.

- "Information Retrieval Assisted Object Segmentation In Video" by Dan Ring and Anil Kokaram in *IET European Conference on Visual Media Production (CVMP)*, London, UK, October 2008.

- "Online Parsing of Sports Coaching Video through Intrinsic Motion Analysis" by Dan Ring and Anil Kokaram in *IEEE International Conference on Image Processing (ICIP)*, San Antonio, Texas, USA, September 2007.

- "Feature-Assisted Sparse to Dense Motion Estimation using Geodesic Distances" by Dan Ring and Francois Pitié in *IEEE Irish Machine Vision and Image Processing conference (IMVIP)*, Dublin, Ireland, September 2009. Awarded Best Paper.

- "Automated editing of medical training video via content analysis" by Kevon Andrews, Dan Ring, Anil Kokaram, Fadel Al Sabah, T. Clive Lee and Cathy Radix in *Proceedings of SPIE, Multimedia Content Analysis, Management and Retrieval*, 2006.

Chapter 2

On Local Image Features

A large part of image processing attempts to derive meaning from images. The desired meaning can belong to different semantic levels. In image processing, features such as colour, edge or motion are regarded as having a semantically “low” meaning, and are usually used to infer “higher” meanings, such as “is a particular object present in the frame?”. This is the typical “bottom-up” approach to semantic understanding of building on low level features to obtain high level information. The problem is that often the “semantic gap” between the low and high level features is too great to be traversed in a single leap.

Local image features, or *feature points*, provide a summary of the image content based on localised functions of image data giving a “middle level” understanding of the image. A common example is the corner detector [70], where corners are detected by finding strong perpendicular image gradients. Although the corners alone are probably not useful, by comparing the local pixel regions around the corners they can be used to identify correspondences between a pair of images. Many applications can be derived from this simple example of correspondence matching between a pair of images, for example image registration tasks [168, 102, 124], video tracking [156, 171], motion estimation [182, 100] and structure from multiple views/motion [32, 101].

Consider an object visible in an image. Any part of the object structure at a detected feature location can be represented by the pixel region surrounding the feature. Assume that a finite set of unique pixel regions or “patches” exist (i.e. distinct corners). It is then possible to represent an object by the collection of indices corresponding to the distinct patches from the dictionary that make up the object. This representation of image data as indices allows for powerful content based understanding throughout images and video. Interesting examples include a “query-by-content” object search in consumer video [161, 162], real-time CD cover recognition from large image databases [125], video object re-detection and shot clustering [152], object detection in web corpora [131] and automatic robot localisation [178].

This thesis focuses on three main areas related to local features, specifically; analysing how feature detector systems can be evaluated, improving matching in difficult images, and using features in previously unrelated applications. The aim of this chapter is to first give the reader some background into local features, explaining why they are useful and how they are typically implemented in modern applications, including a brief history to clarify the local feature development timeline. Later on, the difficulties in establishing correct correspondences are discussed. Finally, the state of the art methods for addressing some of the key issues of local features are presented, highlighting important areas that will appear later in the thesis.

Historical Note

It is difficult to look back at past research and positively identify when the idea of local image features was first proposed. However, their potential importance is usually attributed to the 1976 paper of Marr and Poggio [108]. In their paper, the authors discuss a method by which biological vision systems use point correspondences between images from the eyes to estimate disparity. The authors then propose a computer algorithm for doing the same task, consisting of matching per-pixel grey levels between the image pair. Although the mathematics existed prior to 1976 for estimating depth from an image pair, Marr and Poggio emphasised the importance of the ability to automatically identify correspondences between images would have on these applications.

Between 1977 and 1980, Moravec introduced and developed his simple yet powerful interest point detector (the "Moravec" detector) based on local self-similarity [120]. The idea is to compare neighbouring 8x8 image blocks, and label a region as "interesting" if it is sufficiently different from its surrounding patches. For example, consider a region exhibiting a corner. Looking at the 8 "compass point" neighbours of a corner region, each of the 8 regions will look distinctly different from the corner, and thus the corner will be labelled as an interest point. Although by modern standards this detector is very computationally efficient, requiring only a handful of full-image sum-of-squared differences, at the time this was considered expensive [119]. Another problem is that the detector is not isotropic. Consider an edge that is not aligned along the directions of the neighbours, i.e. if the neighbours are the 8 compass points, and the edge going through the centre region is at a 30° angle. Each of the regions will be sufficiently different to each other and thus the edge is considered interesting. However, if the edge is angled at 45°, many of the regions will be similar and thus not detected as an interest point. Although this sensitivity to relatively small changes in rotation severely reduces the usefulness of the detector, it was still a highly popular feature detector.

Despite computational constraints at the time, research into detecting and matching local features was fruitful, resulting in applications such as camera motion extraction in 1987 [49, 69], recovering the latent 3D structure from a pair of images in 1981 [103], and *model-based* object recognition systems in 1988 [89]. At the same time, Harris & Stephens addressed the non-isotropic problem of the Moravec detector (by orientating the patch to the image gradients in the region), formalising their prolific corner detection function to be featured in research for the next 20 years [70].

What this thesis is not about

There already exists a large field of work involved in matching sets of interest points based solely on point locations, disregarding the image content once the feature locations have been found. In "Shape Matching" literature [58, 16, 123, 18, 122], an object is defined by connecting neighbouring feature locations, i.e. "joining the dots". By enforcing the neighbourhood constraints on the features, one set of connected feature locations is fit onto another set, for example using bipartite graph matching [17] or thin-plate spline matching [24]. Much earlier work by Zahn demonstrates matching sets of point clouds by assuming an implicit point topology, and matches instead the minimal spanning tree of the sets of points [187]. It was commonly believed that the variation between typical pairs of images was too great in many applications, hence the reason the image data is thrown away once the features are computed [13]. While spatial configurations are certainly useful, and will be explored later, the work in this thesis focuses on the use of *appearance* to identify similar content.

2.1 Introduction to Local Image Features

Essentially, local image features are points of interest localised by a function of a low level feature space, such as image pixel gradients. The idea is that images can be represented by these patches of salient regions. The applications that typically use feature points can be divided into two categories; correspondence matching, and content classification. Before comparing these applications, some of the local feature nomenclature to be used throughout this thesis will now be clarified. The terms "local image feature", "interest point" or "feature point" are equivalent and used interchangeably throughout this thesis. The term "object structure" refers to the real-world entity depicted in the image, while the term "image structure" is used to refer to the pixels representing the object structure. The idea of feature points will now be put into context through discussion of the classes of feature based application.

2.1.1 Correspondence Matching

Finding correspondences between images is a key task used by many applications mentioned previously. Motion and disparity estimators are common systems for identifying dense pixel-wise correspondences across different images through time and space respectively [44, 149]. Figure 2.1 shows two typical applications in which feature based correspondence matching is used. The idea is to first *locate* features in each frame and then *match* them between frames. In the multiview application, matches lead to depth estimates, while in object matching scenarios, matches enable object search. The use of local features, as opposed to optical flow or disparity estimation, is becoming increasingly popular for these applications as feature patches by definition have the highest information content in local picture regions. Hence feature based correspondence matching is expected to be robust to varying scene and image conditions. Furthermore, the

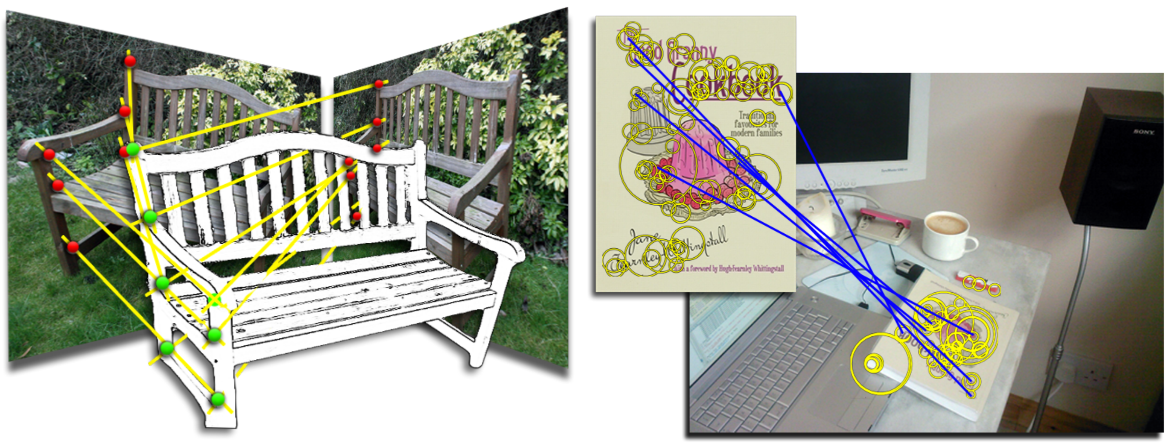


Figure 2.1: Examples of correspondence matching. Left, correspondences between features in a pair of images (red) are used to estimate depth, allowing the shape of the object to be recovered in 3D. Right, features calculated between a query image and desktop scene (yellow circles) are matched (blue) to allow detection and localisation of the book cover despite changes in object scale, rotation and scene illumination.

sparseness of the features leads to computationally efficient systems.

2.1.2 Content-based Classification

When using a content-based classification system, the user presents an input image to the system and asks it to classify the content (object, scene etc.) in the query image using corpus images stored in a database, returning the class of object. Object classification typically uses global (image-wide) features to model objects, such as edge, colour or shape distributions. This works well when the query image and corpus images can be reasonably constrained, for example, if the images contain only a single instance of the object, with the object occupying a large amount of the image and very little background "clutter". Classification accuracy begins to degrade as soon as the image conditions depart from these constraints. The situation is made more difficult if the object in the query image is partially occluded or presented at a different orientation.

To handle reasonable amounts of clutter or changes between query and database images, recent object classification schemes have exploited the ability of local features to find similar content under photometric and geometric warps to great effect [188]. By decomposing the images into patches of salient content, an image can be represented by the occurrence frequencies of the indices corresponding to the patches of the image. This is known as the "bag of words" model [95]. Even under different image conditions, variations of semantically similar content (for example, different instances of a cup) will have very similar sets of patch index frequencies. An example is shown in Figure 2.2.

An interesting extension of this method of object classification is the "query-by-content" ob-

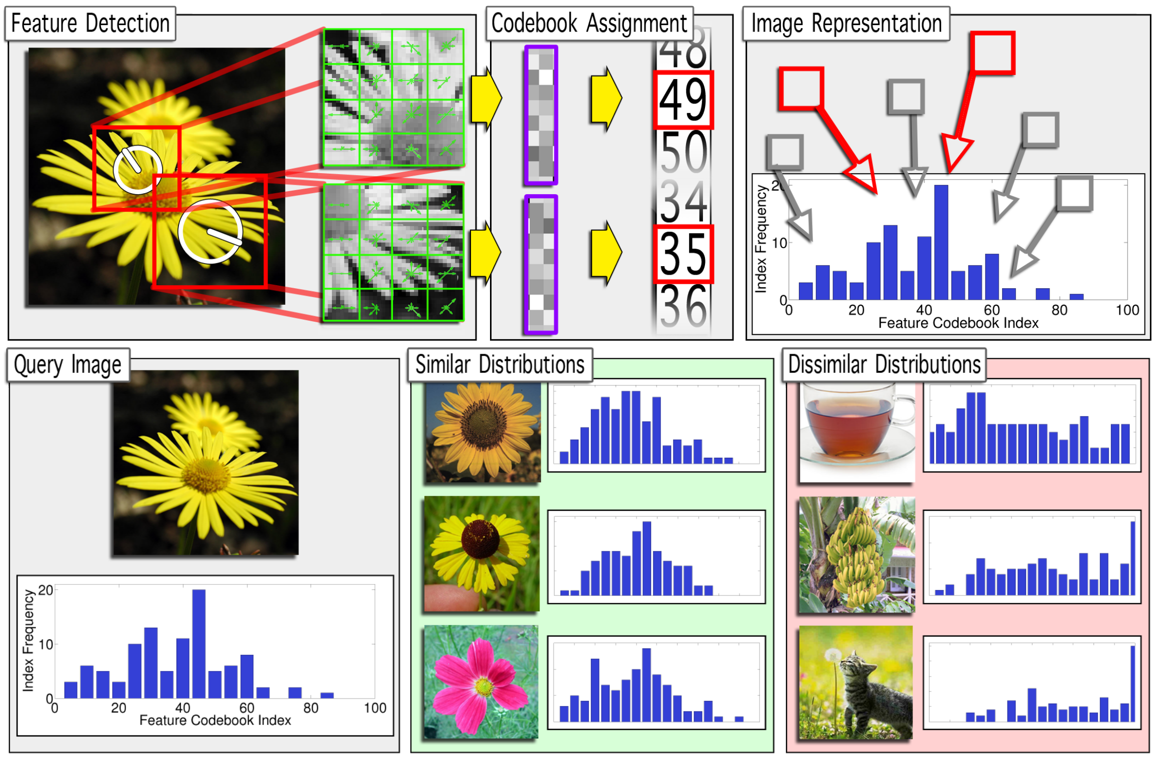


Figure 2.2: Example of "bag of words" model [95] for object classification. Top row, left to right, features are first detected in the image (white circles). Their surrounding pixel regions (red squares) are processed and described by feature vectors (green squares to purple rectangles) that are invariant to varying image conditions. The vectors are quantised, allowing the image patch to be represented by a codebook index (red square). The image is then represented simply by the occurrence frequency of the indices (blue histogram). Bottom row, images similar in feature distributions (i.e. the low level feature space the detector is sensitive to; edge, corner, shape or colour) will result in similar occurrence frequencies regardless of size, pose, illumination etc. as shown by the flower image index distributions (centre). Dissimilar objects will then have different frequency distributions (right). By representing complex images as simple, 1D distributions, object detection in massive image databases becomes a trivial task. For object classification, the computational cost of training a system on a database is significantly reduced, allowing training with larger numbers of object classes, while offering high levels of invariance to different image conditions.

ject detection task (the special case of *one-shot* classification), where each image in the database is classed as belonging to the same class as the query image or not. This is the image-based analogy of a text-based search engine, i.e. the user inputs an image, and the system returns other images from the database similar to the query image. As well as the inherent flexibility offered by local features, another advantage over global features is the ability to localise the detected object from its constituent features. The following section looks at the current state of feature detectors, and how they are being used in modern applications.

2.2 Modern Approaches to Feature Detection

The development of feature points and associated applications since the original Harris & Stephens corner detector [70] has produced faster and more accurate feature detectors, while extending the range of applications using features well beyond direct image correspondence matching. The strategies that feature point applications use typically follow three stages:

1. *Detection*, locations or regions of interest are first found in the image by the feature detector.
2. *Description*, a feature vector is calculated by a function of the pixels in the region around the feature point location, known as a *descriptor*.
3. *Matching*, the feature descriptors are compared to identify similar image content. In an image registration system, the descriptors are used to establish correspondences. In object classification systems, collections of descriptors in a database are compared to those in the target image to detect or localise an object.

One standard source of confusion is the use of the term "feature detector" to mean both the detection and description parts combined as the one system. To make it patently clear to the reader, the term "feature detector" is used in this thesis to refer only to the actual detection stage, with feature description being a separate step.

2.2.1 Detecting Local Features

The method of selecting an image location as a feature point is based on some fixed function of the local pixel region. Given a set of images under varying photometric conditions (image blur, noise, compression artefacts etc.) and geometric conditions (scaling, rotation, translation, perspective distortions etc.), a good detector function is one that can reliably and repeatedly detect the corresponding point of an object structure in all the images of the set. The ability of a detector to find corresponding locations between images at is known as the *stability* (or alternatively the *repeatability*) of the detector, and is independent of the later description or matching stages. The stability of the detector is largely determined by how *invariant* the design of the localisation function is to particular image conditions. In general, the chosen feature space should at least be invariant to translation. For example, consider the "corner space" given by strong orthogonal image gradients. If the corner space is calculated on two images, the second image being a horizontally shifted version of the first, it is expected that the corner spaces will also be translated versions of each other shifted by the same amount. In practice, the corner space alone can also allow for other slight changes between image structures in addition to translation invariance [113]. Modern feature detectors are designed to be invariant to other classes of transformations, the most successful example of which is *scale* invariance.

2.2.1.1 Scale Invariance

The objective of scale invariance is to enable the detector to find the same part of an object regardless of the size of the object within the image. The *scale* of a feature-point is a canonical value that relates the apparent size of one image structure to another. For example, consider a pair of images containing the same object, where the size of the object in the second image is twice as large than it appears in the first. If a pair of features are found at a point on the object in both images, the scale of the feature in the second image should be exactly twice that of the feature detected in the first image.The works of Lindeberg [99], and Bretzner & Lindeberg [30], show how points can be accurately localised in *scale* as well as in the 2D feature space using the Laplacian-of-Gaussian (*LoG*) operator. Local maxima and minima of the function shown in Equation 2.1 correspond to detected scales,

$$LoG(\mathbf{x}, \sigma) = \sigma^2(L_{xx}(\mathbf{x}, \sigma) + L_{yy}(\mathbf{x}, \sigma)) \tag{2.1}$$

where σ^2 is the variance of the applied Gaussian blur, and $L_{xx}(\mathbf{x}, \sigma)$ and $L_{yy}(\mathbf{x}, \sigma)$ are the horizontal and vertical second-derivatives of the blurred input image at the pixel site $\mathbf{x}$. It transpires that detecting local maxima and minima of Equation 2.1 in terms of σ results in detection of the *scale* of the point $\mathbf{x}$ in the image. An example is shown in Figure 2.3. In practical applications, the range of scales, σ, is fixed, and so the scale space of the images can be built by successive smoothing and down-sampling of the original input image. A good description of building the image scale space is given by Lowe [105].

2.2.1.2 Rotation Invariance

It is also useful for a feature detection system to identify similar object structures in the presence of rotation. Typically rotation invariance is incorporated in the descriptor stage, where either a canonical angle value is calculated from the features image structure and used to orient the associated descriptor [105], or the pixels of the image structure are added to a histogram based on their polar co-ordinates removing any angular dependance [188]. A detected feature point, f_p, detected in frame p is typically defined by $f_p = (\mathbf{x}, \sigma, \theta)$, where $\mathbf{x}$, σ and θ are the spatial location, canonical scale and rotation values of the point respectively. To illustrate how modern

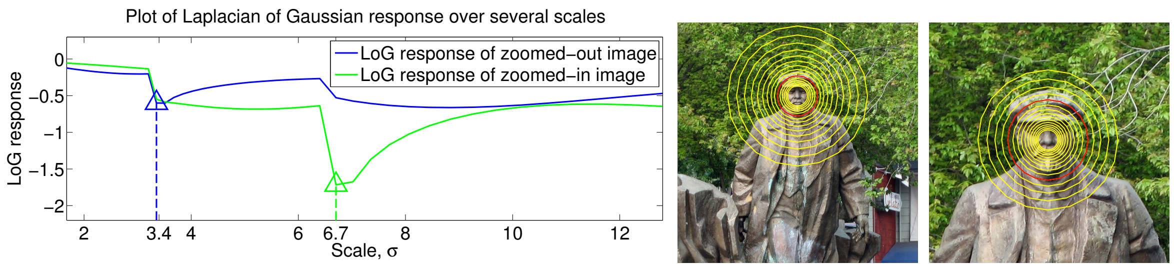


Figure 2.3: Example of using the Laplacian of Gaussian response of Equation 2.1 to localise salient scales for points in an image. On the right are two images, one size being 2x zoom of the other. To illustrate the idea of scale selection, the same point is first manually selected in the two images on the right (the nose of the statue). The right most image is a zoomed-in version of the left image, with an approximate doubling of scale. Left, the *LoG* response is calculated for a range of scales at the point in both images (blue plot for left image, green plot for right image). The scale ranges tested in this example are shown by the yellow circles in the right image, the radii of which are set to 3σ. Note that the same range of scales (i.e. $\sigma = [2, 12]$) is tested on both images. The problem of selecting the filter support size is avoided by using the IIR recursive Gaussian filter of Geusebroeck [65]. In the two plots on the left, the salient scales for this example are given by the first local minima in the two plots, shown by the triangles. These detected scales are shown by the red circles in the right image pair. Notice that the difference in detected scales using the *LoG* response plots is approximately two-fold, corresponding to the change in image scales.

feature detectors exploit characteristics of low level features to localise features in an image, a selection of detectors are now presented.

2.2.1.3 Harris-Laplace

The Harris-Laplace detector of Mikolajczk & Schmid [115] applies a multi-scale framework of the original Harris & Stephens corner detector [70] to locate feature points. Corners offer excellent spatial localisation due to their property of strong quasi-perpendicular gradients. A good way of encoding image gradients for this purpose around a point $\mathbf{x}$ is given by the second-moment matrix:

$$\mu(\mathbf{x}, \sigma_d, \sigma_i) = \sigma_d^2 g(\sigma_i) * \begin{bmatrix} L_x(\mathbf{x}, \sigma_d)^2 & L_x(\mathbf{x}, \sigma_d) L_y(\mathbf{x}, \sigma_d) \\ L_x(\mathbf{x}, \sigma_d) L_y(\mathbf{x}, \sigma_d) & L_y(\mathbf{x}, \sigma_d)^2 \end{bmatrix} \tag{2.2}$$

where $L_x(\mathbf{x})$ and $L_y(\mathbf{x})$ are the horizontal and vertical image gradients at every pixel site $\mathbf{x}$ over a range of scales σ_d. The scale σ_i is used in the Gaussian kernel $g(\sigma_i)$ is known as the "integration scale", and relates to σ_d by a constant s, typically $s = 0.7$, i.e. $\sigma_d = s\sigma_n$ [115]. The purpose of applying a second smoothing at the integration scale is to remove any aliasing effects from the products of the gradients at the differentiation scale.

The strength and orientation of a corner at an image patch, centred at $\mathbf{x}$ and smoothed by σ_d and σ_i according to Eq. 2.2, is related to the eigenvalues of $\mu(\mathbf{x}, \sigma_d, \sigma_i)$. To avoid having to explicitly calculate the eigenvalues of μ, it is noted that the determinant and trace functions of

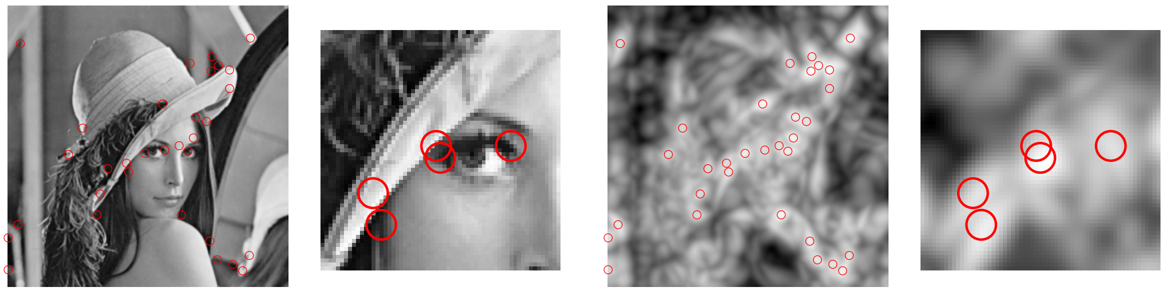


Figure 2.4: Example of Harris-Laplace points taken at a fixed scale of $\sigma_d = 4$ and $\sigma_i = \sigma_d/0.7$, and corner threshold of $C > 0.001$ (Assuming a gray-scale input image with intensities in the range $[0, 1]$). Original and corner space C images are shown by the left and right image pairs.

matrices are related to the eigenvalues; the determinant is the product of the eigenvalues, the trace is the sum. The corner space C for a point $\mathbf{x}$, at differentiation and integration scales σ_d and σ_i is calculated using the modified Harris corner measure of Noble [126], e.g.

$$C(\mathbf{x}, \sigma_d, \sigma_i) = \frac{|\mu(\mathbf{x}, \sigma_d, \sigma_i)|}{\text{trace}(\mu(\mathbf{x}, \sigma_d, \sigma_i)).} \tag{2.3}$$

where $||$ refers to the determinant operator. The maxima of $C(\mathbf{x}, \sigma_d, \sigma_i)$ indicate the presence of strong bi-directional edges, which are detected as feature point locations. As per [115], the feature-point scale σ_n is then selected by the *LoG* detector, as outlined in Equation 2.1. Using σ_n as the input scale ($\sigma_i = \sigma_n$ and $\sigma_d = s\sigma_i$), the corner space C is recalculated, and a new maximum in C in the vicinity of the previous feature is redetected. This process iterates until convergence where a maximum of both corner- and scale-space is achieved. An example using a fixed detection scale is shown in Figure 2.4.

2.2.1.4 Wavelet-Based Detector

Fauqueur et al. propose a feature-point detector using the Dual-Tree Complex Wavelet transform (DT-CWT) [50, 22, 83]. Using the products of the 6 oriented sub-band coefficients f_k, a "junction space" is calculated to emphasis corners at the scale (wavelet level) l

$$\begin{aligned} P^{(l)} = & \quad \alpha^l \left(\textstyle\prod_{k=1}^{6} |f_k|\right)^{1/6} \\ A = & \quad \textstyle\sum^L g_l(P^{(l)}) \end{aligned} \tag{2.4}$$

where α is the weighting of each wavelet level, and L is the number of levels. The multi-scale junction space A accumulates $P^{(l)}$ over each wavelet scale $l \in \{1, \ldots, L\}$ using a 2-D interpolation function g_l. Feature-point locations are detected about the maxima in the resulting A. As the wavelet decomposition is inherently a scale space representation, both Fauqueur et al. [50] and Bharath & Kingsbury [22] provide proprietary scale selection methods. An example of features detected using the DT-CWT detector is shown in Figure 2.5.

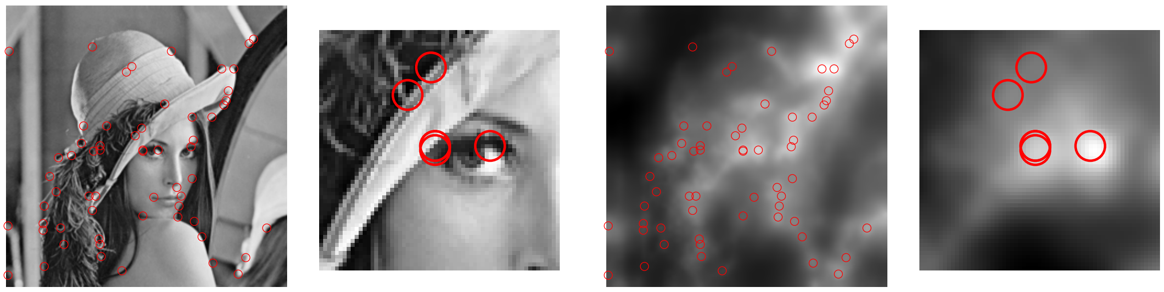


Figure 2.5: Example of DT-CWT feature points using parameter values $\alpha = 0.1$, $\beta = 1/6$, and across all valid scales. Original and wavelet space A images are shown by the left and right image pairs.

2.2.1.5 SIFT

The Scale-Invariant-Feature-Transform (SIFT) is a popular feature detector system developed by Lowe, described in detail in [104, 105]. To put the SIFT detector in context with the other detectors discussed so far, a brief overview of the SIFT detector is given, however the reader is directed to the original works by Lowe, or the works of Wu [184] and Vedaldi [175] for implementation specifics and practical considerations. The SIFT detector begins by using a fast approximation of the LoG operator, the Difference-of-Gaussian (DoG) operator, at multiple scales to give a 3D volumetric feature space, $D(\mathbf{x}, \sigma)$. Feature points are detected by extrema in the space $D(\mathbf{x}, \sigma)$, refined to sub-pixel accuracy in the spatial dimensions, and to sub-scale accuracy in the scale dimension. A candidate feature is tested for a strong corner response, calculated by Equation 2.3 (for the single pixel site only) and rejected if it falls below a threshold. One of the advantages of the SIFT detector is that the scale and location of the feature are localised simultaneously in the same feature space, as opposed to Harris-Laplace iterating between corner space and LoG space for example. This results in detected features that are more stable in both scale and location. An example of features detected by the SIFT detector at a fixed scale is shown in Figure 2.6.

2.2.1.6 MSER

Maximally-Stable-Extremal-Regions are based on the idea of level-sets, introduced first by Matas et al. [111]. In this context, a level set at a particular threshold t is defined as the spatially connected pixel regions in a binary image given by $[I > t]$, where I a gray-scale image and $[\Phi]$ is the indicator function which is 0 if the condition Φ is false, and 1 otherwise. A "maximally stable region" is a region in which the number of connected pixels for the region do not change over a number, δ, of threshold values. The reader is directed to the work of Matas et al. or Vedaldi [175] for a better understanding of level-sets and MSERs.

The use of these level sets has been extended by [56] and [57] to incorporate multi-scale

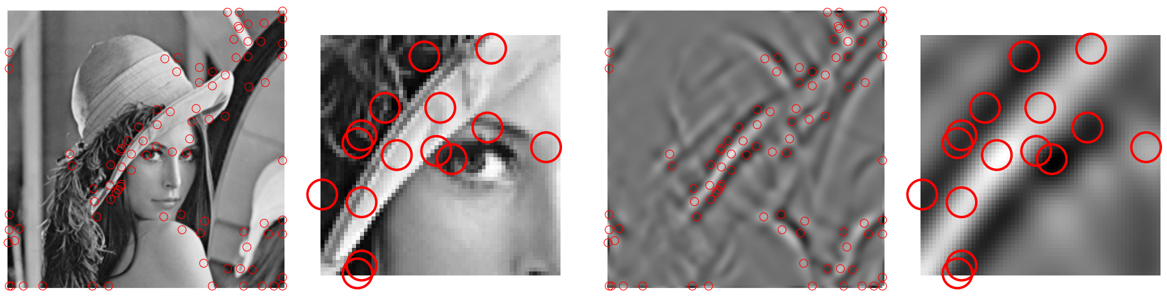


Figure 2.6: Example of SIFT feature points at a fixed scale of $\sigma_d = 3.5$, and a threshold of $DoG > 0.00002$ (Assuming a gray-scale input image with intensities in the range $[0, 1]$). Original and DoG space images are shown by the left and right image pairs.

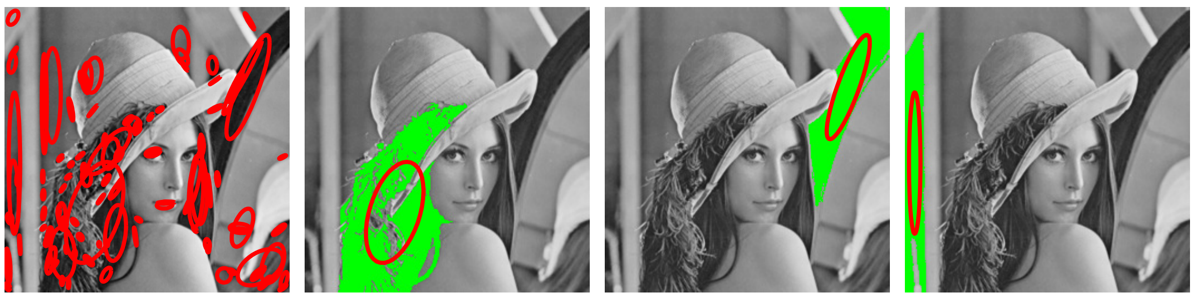


Figure 2.7: Left, example of all ellipses detected on the image, using a connectivity tolerance of $\delta = 2$. Images two to four show the three largest detected connected components in the image and their associated ellipses (the MSER implementation demonstrated here is that of Vedaldi [175]).

frameworks and additional features such as shape and colour. MSER uses an intuitive approach of pixel sorting and region comparison to segment contiguous regions. A pair of neighbouring pixels are considered contiguous if the difference between the pixel intensities is within a tolerance δ. Ellipses are fitted to the detected regions using the covariance of the pixel locations. Feature-point locations and scales are extracted directly from the ellipses. In general feature descriptors are calculated on square or circular regions around the detected feature location. As MSER detects elliptical regions, an adaptation to the descriptor function is sometimes made to pre-warp the data in the local ellipse, as demonstrated by Forssén & Lowe [57]. An example of the regions that MSER detects and the ellipses calculated on them is shown in Figure 2.7.

2.2.1.7 Hessian-Laplace

The Hessian-Laplace detection function is simply the scale-weighted determinant of the Hessian matrix

$$H(\mathbf{x}, \sigma) = \sigma^2 \begin{vmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{vmatrix} \tag{2.5}$$

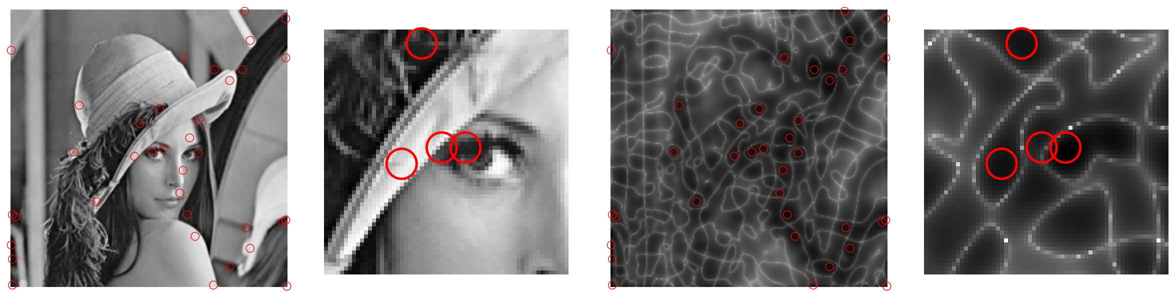


Figure 2.8: Example of Hessian-Laplace points taken at a fixed scale of $\sigma_d = 6$, and corner threshold of $H > 0.0001$ (Assuming a gray-scale input image with intensities in the range $[0, 1]$). Original and Hessian space H images are shown by the left and right image pairs.

where $||$ refers to the determinant operator. As it is based on the second differential, it is sensitive to blobs and ridges.The Hessian-Laplace detector is similar in structure to the LoG scale selection method, as the trace of H corresponds to the LoG detector (Equation 2.1). As with the Harris-Laplace detector, the feature-point location and scale are iteratively refined.

2.2.1.8 Other Detectors

The detectors that have been discussed were chosen based on their novelty of feature space and popularity in contemporary literature. However, it is worth nothing that there is a staggering selection of other detectors and variants of the ones listed above [166, 79, 4, 129, 174, 139, 51, 56, 36, 98, 107, 63, 176, 147, 37]. The following discussion section acknowledges some of the more interesting derivative work in feature detection, however the works are not strictly relevant to this thesis.

Often feature detection systems are computationally expensive. Some feature detector systems tailor existing detectors for a specific purpose, allowing certain elements of the system to be sacrificed to improve overall computation speeds. The SURF ("Speeded Up Robust Features") detector is an optimised version of the multi-scale Hessian detector, using integral images to generate the feature multi-scale feature space, and a descriptor and matching system tailored specifically for image registration tasks [14].

One interesting detector is the "Fast corner detector" of Rosten et al. [147]. The corner detector proposed by Rosten et al. reduces the problem of corner detection down to a direct comparison of pixels sampled in a circle at equal angle spacing and fixed radius around a candidate site, to see if the site "looks like" a corner. The idea is that if enough of the intensities of the circle pixels are greater that the intensity of the candidate pixel, the site is likely to be a corner. Additional tests are required to improve detection accuracy, and there are a number of caveats, however, the speed of feature detection is significantly faster than other detectors.

Hardware can also be exploited to realise significant performance gains. By adapting detectors to operate on the graphics processing unit (GPU), feature detection on PAL resolution images can be performed in real-time 25 fps for the purposes of video tracking [159, 184]. This makes sense, as most stages of feature detection algorithms can be readily partitioned for parallel execution on the the highly parallelised GPU pipeline.

Some detectors extend the work of existing detectors to provide added invariance to general or application specific image conditions. Mikolajczk & Schmid [115] extend the Harris and Hessian detectors to provide greater invariance to local affine warps of the object structure by iteratively adapting the elliptical shape of the image structure to achieve isotropic corner gradients. Although results show dramatic increases in the number of correct matches identified between images undergoing large geometric warps [117], the computation time required to identify the features is infeasible for most practical applications. Other work in extending the level of invariance for specific application image conditions shows the localisation of useful features in both space and time. For example, by observing video as a volume and looking for “T” junction features across spatio-temporal “slices” to detect occlusions [7], or simply extending traditional 2D detectors to 3D volumes [124].

2.3 Describing Local Features

When matching by “appearance”, correspondences between a pair of features are found by comparing pixel information around the feature sites. Marr and Poggio [108] originally suggested simply comparing the image intensities at the single pixels at the sites of the interest being evaluated. Determining matches in this fashion makes it difficult to identify correct matches for two reasons: firstly, the correlation between neighbouring pixel intensities observed in typical images means single pixel intensities are not suited for uniquely identifying correspondences. Secondly, any change in illumination between images will induce changes (uniform or otherwise) in pixel intensities.

A better idea is to compare many pixels around the sites of candidate feature matches. For example, at each feature point in a pair of images A and B, assemble a feature vector comprised of the 5 x 5 patch of pixel intensities centred around the feature point site. A feature match is then established between a pair of features in A and B if the feature vectors (descriptors) are sufficiently similar. The simplest way to compare a pair of descriptors is to simply calculate the mean squared error (MSE), and accept the match if the MSE is below a threshold. The averaging effect of comparing patches of pixels in this way allows greater invariance to image noise than simply comparing single pixels. Using normalised cross correlation (NCC) as the comparison measures provides some additional invariance to patch-wide changes in image intensity (i.e. from illumination changes), as the intensities in the feature vector patches are normalised prior to comparison. However, other than slight image noise and changes in intensity, any changes in image structures of the feature patches (such as a small rotation or scale change) will result in large values in the MSE, and values close to zero for the NCC distance measures. A good descriptor should be invariant to a wide range of photometric and geometric conditions.

As with feature detectors, there is a variety of descriptors to choose from. For example, in their studies of local features for texture classification, Zhang et al. [188] and Lazebnik et al. [90] present two complementary feature descriptors, "RIFT" (Rotation Invariant Feature Transform) and "SPIN", both designed specifically for high invariance to patch rotation. The SPIN detector is a two dimensional histogram of pixel intensities in a region, indexed by the pixel intensity, and the spatial distance from a pixel in the region to the region centre. The RIFT descriptor breaks the feature image patch into (four) concentric rings. Each ring has an 8 bin orientation histogram that accumulates the magnitudes of the local pixel gradients, indexed by the orientations of the pixels. In other descriptor research, Winder et al. [183] propose a modular framework for descriptor design, and use a ground truth to learn the optimum design choices and associated parameters.

The SIFT descriptor is probably the most prolific contemporary feature descriptor, due in part to its grounding in biological science, its impressive all-round performance [116], and the availability of reference [105], adapted [175] and optimised [159] source code. As the focus of this thesis is on using features and not on the design of detectors and descriptors, the SIFT detector will be used as the default descriptor throughout this thesis. An introduction to the SIFT descriptor is now presented.

Research by Edelman et al. [46] discussed how the human visual system (HVS) interprets real-world objects by a hierarchy of many "simple", oriented functions (Gabor-like filters) to "complex cells", and how an analogous system could be created for artificial object recognition. From an image processing point of view, the simple cells are sensitive to the location and orientation of gradients, and are only slightly invariant to changes in the apparent image structure. The complex cells are made up of the responses from many overlapping simple cells belonging to nearby orientations and spatial locations. Although dramatic changes in the image will "activate" different simple cells, the same sites in the complex cell are activated by both the changed and original object structures, effectively enabling greater invariance to image conditions.

In his work, Lowe [105] extends the work of Edelman et al., developing his "SIFT" feature descriptor based on the idea of simple and complex cells. The descriptor function identifies the image patch surrounding the feature as a multiple of the canonical scale, σ. The patch is then decomposed into sub-patches, which are the complex cells. Each complex cell is a gradient histogram. The simple cells are then given by the magnitudes and orientations of each of the local pixels in the large image patch. The contributions of the magnitudes of the simple cells are distributed among multiple spatially nearby complex cells, analogous to the overlapping simple cells in the biological model. The entries into the complex cell histograms are given by the orientations of the simple cells, an example of which is shown in Figure 2.9. The local image patch is typically partitioned into $4 \times 4 = 16$ spatial regions, each region having an 8 direction orientation histogram. Concatenation of the histograms of the 16 regions results in a $16 \times 8 = 128$ element feature vector. To provide additional illumination invariance, the vector

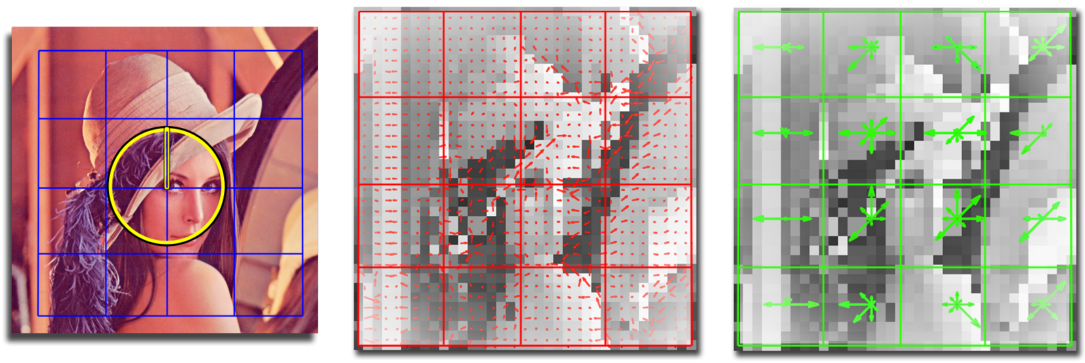


Figure 2.9: Left, a single feature has been detected, and will now be described by a SIFT descriptor. The radius of the support region for the descriptor (blue) is a little over twice the canonical scale of the detected feature (yellow circle). Centre, the magnitudes and angles of the local pixel gradients ("simple cells", red) are collected, with the contribution of each pixel weighted by a Gaussian window centred on the feature point. Right, 4 by 4 local histograms are computed (green arrows), indexed by the orientations (quantised to 8 bins) and weighted by the magnitude of the local gradients in the supporting "complex cell" (green squares). The histograms are then normalised and concatenated to give the $4 \times 4 \times 8 = 128$ element feature vector.

is normalised such that its L^2 is 1.

2.4 Difficulties in Identifying Correct Feature Matches

The research in local feature points has over the years resulted in a large selection of detectors and descriptors to choose from. The interesting task now is how to leverage these detectors and descriptors to identify correct correspondences. In practice there are two issues to be resolved; firstly, given a feature detector / descriptor pair, how can matching points be found. Secondly, what is the best detector / description combination for a given application.

2.4.1 Finding Matches

Assuming that the chosen feature detector has found useful points of interest, and the descriptor algorithm has represented the image structures at the feature sites with a feature vector, the goal now is to to use the descriptors to find corresponding image patches. The task of matching relevant to the proposed work can be broadly separated into two categories; correspondence matching and matching against features in a database. The first involves estimating correspondences between an image pair where the image content is expected to be the same in both images, but has undergone some transformation, i.e. perspective warp between images from multiple cameras, or motion between consecutive frames in a video sequence. The second category belongs to the task of matching feature vectors from a database to a target image, for example, in the *model-based* object recognition scenario.

In a typical correspondence matching scenario, it is required to match content from a source image P to content in the target image Q. Feature points are calculated on the pair of images P and Q, with the sets of the interest points denoted, F_p and F_q. The descriptors belonging to the feature sets are denoted D_p and D_q. For each feature $f_p \in F_p$ (and associated descriptor $d_p \in D_p$), the goal is to find the corresponding feature $f_q \in F_q$ (and $d_q \in D_q$) that has the same appearance, i.e. the descriptor d_q that is most similar to d_q. The set of the feature correspondences is given simply by the nearest neighbour in the descriptor space, using a traditional distance measure such as the L^2, χ^2, or Earth Mover's Distance (EMD) [188],

$$\begin{aligned}
\mathrm{NN}_{L^2}(f_p, F_q) = & \quad \operatorname*{argmin}_{f_q \in F_q} \sum_{i=1}^{N_d} (d_p(i) - d_q(i))^2 \\
\mathrm{NN}_{\chi^2}(f_p, F_q) = & \quad \operatorname*{argmin}_{f_q \in F_q} \frac{1}{2} \sum_{i=1}^{N_d} \frac{(d_p(i) - d_q(i))^2}{d_p(i) + d_q(i)}
\end{aligned}$$

where N_d is the number of elements in the feature descriptors d_p and d_q (i.e. 128 for SIFT descriptors). The calculation of the EMD is more complicated, the reader is directed to Zhang et al. [188] for a clear definition of how the EMD applies to feature vectors. Incorrect matches are rejected if the descriptor distance exceeds a threshold. Using an exhaustive search strategy to find the nearest neighbours for example, each feature $f_p \in F_p$ is compared with every $f_q \in F_q$. The f_q having the minimum distance to f_p w.r.t. feature vectors is selected as a correspondence. Recall that a feature detector may detect several thousand feature points on a single image, and that the size of the descriptor for each point is reasonably large (i.e. SIFT descriptors typically have 128 elements). Hence given $[F_p \times F_q]$ possible matches, exhaustive search can take a long time. In some matching applications, the number of candidate matches can be reduced, for example, it might be known that the maximum translation between sites of a pair of features is limited to a fixed distance, candidate matches with a translation greater than this value are rejected. Similar conditions may apply to rotation or scale values, θ or σ. However, in the general matching case, no such prior knowledge can be applied, and the matching task remains expensive.

The nearest-neighbour scheme on its own is sufficient only for very simple matching scenarios where the difference in content between the pair of images being matched is extremely low. Usually this scheme poses a number of difficulties in correspondence matching. A good matching scheme minimises the number of incorrectly assigned matches when the underlying object structures are different (Type 1 error), and also minimises the number of missed matches when the object structures are the same (Type 2 error). The easiest way to lower the Type 2 error is to ensure the chosen feature detector is reliable at re-detecting features in the presence of challenging image conditions between the pair. However, an ideal feature detector does not exist, and so the amount of Type 2 error largely depends on the degree of variance between the images being compared. In practice, Type 1 error is the more difficult problem faced in the matching problem. These error types can be clearly observed in Figure 2.10. Type 1 error

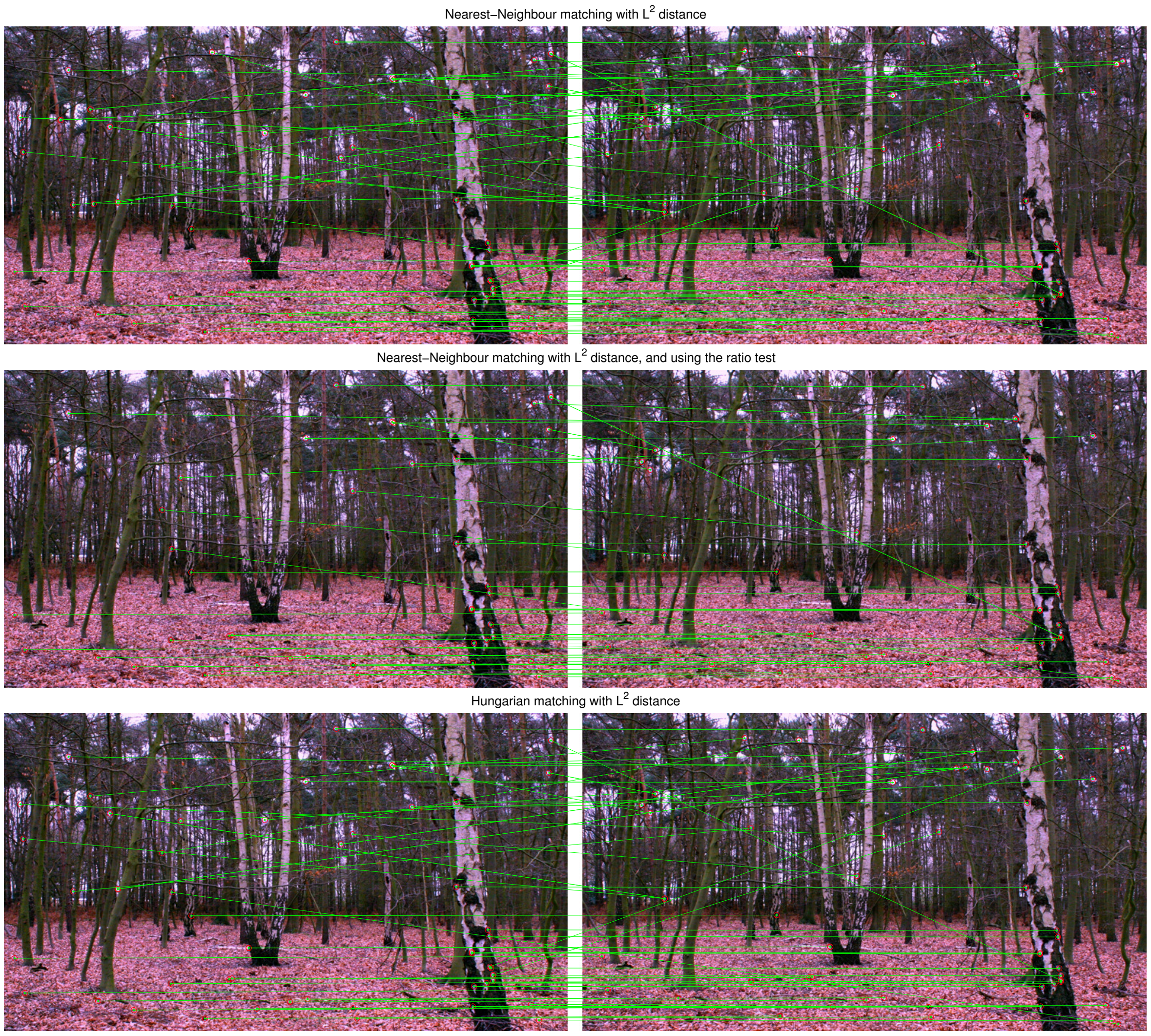

Nearest−Neighbour matching with L2 distance
Nearest−Neighbour matching with L2 distance, and using the ratio test
Hungarian matching with L2 distance

Figure 2.10: Difficulties in matching features. The descriptor comparisons are calculated using the L^2 distance. To easily verify matches by inspection, the example images have been pre-registered, and should be separated by a horizontal translation only. Correct matches should therefore appear as horizontal blue lines, with incorrect matches as diagonal lines. Top, nearest neighbour matching with a matching threshold of 0.2. Middle, nearest neighbour, again using a threshold of 0.2, but also rejecting matches if the ratio of nearest to second-nearest neighbour distances is greater than 0.8. Bottom, using Hungarian matching [5] to solve the linear assignment problem for a global minimum matching cost, rejecting matches above a match threshold of 0.2.

manifests as the diagonal (incorrect) matches, while the effect of Type 2 error is noticed by the lack of any matches (correct or otherwise) in certain areas of the images.

False alarms in matching arise from two reasons; firstly, if part of the object is present in P but not in Q (occlusion), and there exists another image structure in Q with sufficiently similar appearance to the features being matched from P. Secondly, a feature not belonging to the object in Q has a lower descriptor distance than the true feature that does belong to the object, simply by chance. This false alarm matching is exacerbated by the level of invariance designed into the chosen descriptor; by increasing the the descriptor invariance, dissimilar image patches are more likely to produce similar descriptors. A heuristic method proposed by Lowe [105] attempts to mitigate this one-to-many matching problem, by enforcing the constraint that a match will only be considered if the ratio between the descriptor distances of the first- and second-best nearest neighbour matches exceeds a certain threshold. This ratio is effectively a measure of how distinctive the source feature being matched is. An example is shown in Figure 2.10 (middle), notice that although the overall number of matches is lower, the proportion of correct matches is much higher than just nearest neighbour matching (top). Although this is useful in some situations, methods are needed not only to remove incorrect matches, but to also accept more correct matches.

Using an energy minimisation scheme is a good approach to improving feature matches, i.e. by assigning costs to match configurations and trying to solve for a global minimum cost. Consider assigning the cost of a match between a pair of features as the distance between their descriptors. The match assignment with the global minimum cost is then given by solving the linear assignment problem. The linear assignment problem is a special case of the "minimum cost flow" problem in graph theory and can be solved by setting up and solving the appropriate graph, as discussed by Torresani et al. [173]. The "Hungarian matching" algorithm is simpler to understand and can also be used to solve the linear assignment problem and find the best global match configuration. Although it is easier to understand, it is still involved, and so for more information on this problem and the Hungarian matching algorithm, the reader is directed to the introductory text to linear algebra by Anton & Rorres [5].

An example of using Hungarian matching is shown in Figure 2.10 (bottom), notice that the majority of matches are similar to those found simple nearest neighbour matching (top). Often nearest neighbour matching with a sufficiently low distance threshold will provide a matching configuration with cost very close to the global minimum. In such cases, the linear assignment problem provides limited benefit at a significant computational overhead.

From the false alarm and missed matches observed in the relatively simple matching task shown in Figure 2.10, it is clear that appearance alone is not sufficient for identifying a reasonable number of correct matches. While additional heuristics, such as the ratio test, are quick and often effective in many situations, it does not significantly improve the difficult task of feature matching. The use of an energy minimisation framework for feature matching is still interesting, and is discussed in detail later on.

2.4.2 Selecting Detectors & Descriptors

Having selected a matching strategy, the next issue is selecting the appropriate feature detector and descriptor functions for the given task. The desirable key factor in any feature detection system is high *stability*; the ability to re-detect features in corresponding images regardless of image conditions. However, measuring the stability of the detection system is not trivial. As noted by Rosten et al. [147], there has been more interest in developing new detectors than comparing existing ones, making the selection process more difficult. The lack of comparative studies is not surprising considering the number of detectors and descriptors that exist, in addition to the many, sometimes delicate, associated parameters. Although detector and descriptor evaluation through empirical successes or failures of feature-based applications is interesting, no meaningful contribution can be attributed to the detector or descriptor as separate from the rest of the system. Instead, the need for systems to evaluate and compare detectors and descriptors in an objective, systematic manner is vital in the design and development of both applications, and future detectors and descriptors.

In a real-world scenario, the developer of a feature-based system needs to know what detector is the best suited for the given purpose[1]. One solution is to first generate some ground truth matches on a set of application images, subsequently swap in and out detectors and descriptors while detecting and matching points, and calculate the matching accuracy against the ground truth. For example, in a video tracking application, explicitly define a number of track paths over a sequence of frames, then calculate tracks from a set of detectors and measure how close the estimated tracks adhere to the manually specified ones. This approach is good, in the sense that domain specific data are being used as the quality measure to make vital system design decisions. On the other hand, the user is making an implicit assumption that the images used in the ground truth are representative of images expected to be seen by system. Otherwise, the results are only valid for those ground truth images. This can be remedied by creating more ground truth on more images, thereby increasing the representative set of images.

Although creating and using ground truth is good practice and provides the user with a relevant, application specific measure of detector quality, it is time consuming to create a sufficient number of useful, accurate ground truth matches. In some circumstances, the user can opt to use ground truth created by semi-automatic processes, for example, in a multiple view registration setting, a rough ground truth is found by fitting a perspective transform to detected points by enforcing epipolar geometry constraints [121]. However, semi-automatic scenarios like these do not arise often, and bring with them their own caveats. In some cases, the problem is not how to acquire ground truth, but how to use it efficiently. In object classification, the number of example images (ground truth) used to train the system rises with the number of

[1]The "given purpose" may have additional practical requirements such as computational efficiency, memory or time constraints, source code availability or licensing issues etc. However, it is assumed in this scenario that the user is only interested in a high feature redetection rate.

desired object classes. At some point when the number of classes is high, the time required to train the system becomes too long, making the trial and error method of swapping in and out detectors and descriptors, or tweaking parameters infeasible.

When interpreting feature matching performance from comparative studies, care must be taken to recognise the level and scope of the evaluation study. For example, early work on evaluating performance of corner detectors used contrived image situations containing relatively few corners, Rajan & Davidson [137], or synthetic images, Cooper at al. [37, 36]. These comparative studies were useful in determining the idiosyncrasies of basic corner detectors, for example, the limits of what is considered a corner, and the position of the detected corners on the image structures. In modern applications the demands on feature detectors is much greater, with the focus of detector performance shifted from low-level questions, such as "Can the detector find T-structures?", to higher-level questions, such as "How many correct matches does the detector find given a pair of images undergoing transform X?". Comprehensive studies have sought to test many detectors and descriptors under many transforms "X" [117, 116], seeking to provide insight into the overall behaviour of the chosen detector or descriptor from sets of mini tests. This approach is interesting as it is attempts to provide an application independent measure of detector performance. The caveat is that the sets of tests performed are only useful if they represent image conditions expected in the actual application. Typically, only a limited set of the transforms "X" will be specific to the application, i.e. perspective change and motion blur in video tracking, object orientation and compression artefacts in image classification, and of those, the actual transform conditions will have their own unique variations. Additionally, although it is known that different detectors can be used to complement each other [117, 121], no comparison method has demonstrated how to measure the effects of combining detectors or transforms.

To summarise, how relevant a feature detector and descriptor evaluation system is to the user depends on how much faith the user has that the experimental conditions are representative of actual image conditions expected to be presented to the system. In the absence of any foreknowledge, the user is brought back to the specificity problem of performance measures; "application specific" versus "application agnostic", the choice of whether to develop and test against manual (or semi-manual) ground truth on highly relevant application images, or to use the results data of a broad, comprehensive study into general feature detector and descriptor behaviour.

2.5 Addressing Issues in Practical Feature-based Systems

In this section, state of the art methods for identifying correspondences, and measuring feature detection performance are presented. Some parts are described in more detail than others as they are relevant to later work in this thesis. It was noted that correctly matching features using descriptors alone is difficult for two reasons; the case when detectors do not find the same corresponding point in the next image, and the case where ambiguous matches exist due to multiple similar descriptors in the same image, causing false alarms. In general, the first issue is due to either the level of invariance of the detector or the actual image conditions, remedies of which are outside the scope of this thesis. Regarding the second issue, there are many situations when false alarms can be mitigated by incorporating additional information, in particular, the spatial arrangement of the points themselves.

2.5.1 Incorporating Spatial-Context

One way of reducing the number of false alarm matches is to consider the spatial contexts of the feature points. To identify a spatial context, the question to ask is how do features relate to each other? In image segmentation tasks, the simple Potts model [135, 29] can be used to create a Markov Random Field, where the labelling of a pixel should be consistent with that of its neighbours. However, because of the sparse and irregular distribution of feature points, the notion of a neighbourhood is unclear.

The works of Moreels & Perona [121] and Schmid & Zisserman [150] fit a model such as an homography or epipolar camera transform to best describe the apparent warp as given by initial, tentative feature descriptor matches. The goal is to find an agreement between groups of features that undergo the same transformation. For instance, in an video object tracking example, one group of features might belong to the movement of the object, while another group might belong to the background. Feature matches that do not agree with these two dominant models are rejected. The models are fit by “Random Sample Consensus” (RANSAC) as proposed by Fischler & Bolles [53], where putative transform models are estimated from random subsets of features, and evaluated on other random subsets of the data. The algorithm selects the transform that gives the highest agreement (minimum sum of squared distance) between the feature points in the second image, and those projected from the first image into the second using the estimated transform. Matches that do not conform to the transform model are rejected. Using only a single transform model assumes that the entire object is rigid, and will often reject good matches relating to the non-rigidly deformable parts of the object. This can be accounted for by using more than one transform model to describe the apparent object motion, however, success is highly dependent on correctly estimating the number of models [172].

In recent work, Lombaert et al. [102] use correspondences between features to boot-strap a “sparse to dense” image registration system, i.e. propagating information from sparse features to the dense, uniformly sampled pixel grid of the image. Using feature matches in this way allows for a reduced set of candidate disparities to be identified based on the distribution of disparities from the feature matches, including potentially large disparities that would otherwise be neglected by the typical uniform candidate set (i.e. a 15 x 15 grid). To propagate information from the feature matches to the pixel level, Lombaert et al. encourage all pixels within a fixed L^2 distance of a feature match to have the same displacement as the match. The results of

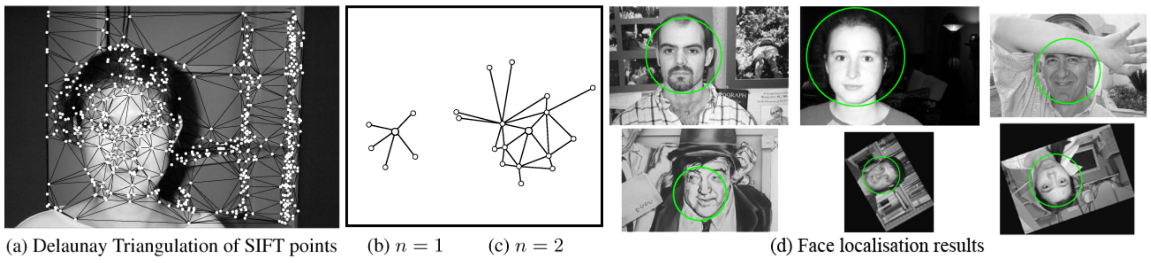

(a) Delaunay Triangulation of SIFT points
(b) $n = 1$
(c) $n = 2$
(d) Face localisation results

Figure 2.11: Example of using Delaunay triangulation to establish a feature neighbourhood. (Images courtesy of O'Regan & Kokaram [128])

Lombaert et al. are interesting, and show successfully registered images despite difficult image conditions. Ring & Pitié [42] extend the idea of propagating feature match information to nearby pixels, but instead of using the L^2 distance, the geodesic distance is used which takes into account the image topology between features and pixels. For example, the distance between a pixel and nearby feature point will be high if a large gradient exists between the two, and low otherwise. The improvement over the L^2 distance is most noticeable in situations involving motion boundaries, where matched features on a moving foreground object should influence only those pixels enclosed by (and belonging to) the object for example. Although the works of Lombaert et al. and Ring & Pitié are more related to feature-to-pixel than feature-to-feature relationships, they show that exploiting the spatial neighbourhood constraints of feature points can produce interesting results.

O'Regan & Kokaram [128] present a novel approach to defining a feature neighbourhood by calculating the Delaunay triangulation of all features in the image, as shown in Figure 2.11 (left). How "close" one feature is to another in the neighbourhood is given by the level of connectivity (i.e. the number of edges) between the pair of features, examples of features connected by 1 and 2 edges are shown in Figure 2.11 (centre). Consider a set of images with different content, but containing the same object, such as images of different faces. The idea is that the although different images will produce different triangulations, the general spatial neighbourhoods of detected features will be similar. For example, features detected on a person's mouth will likely contain links to the persons nose and jaw line. O'Regan & Kokaram present an energy minimisation framework for classifying features as belonging to a persons faces, and subsequently localising the face in the image, even in cases of partial occlusion and geometric warps, examples of which are shown in Figure 2.11 (right).

In their work on model-based database object matching Sivic & Zisserman [162, 161] use a simple heuristic approach using nearby feature points to deal with false positive matches. Consider a potential match between a feature f_p in image P and f_q in image Q, with the sets N_p and N_q comprising the nearby features within a fixed radius of f_p and f_q respectively. The constraint of Sivic & Zisserman requires that at least N similar matches exist in the vicinities of f_p and f_q, i.e. that at least N matches must exist between the features in N_p and N_q before the match between f_p and f_q is accepted. This simple constraint is very effective in removing incorrect matches, but is a post hoc approach; it can only remove bad matches after the matching step has been performed. If a match is incorrect, it does not suggest alternative matches.

Another way of imposing a spatial context is to model the relative locations from each feature point to an arbitrary point in the image, known as a *star-graph*. This probabilistic model has been used by Shechtman & Irani [154] and Boiman & Irani [23] for accurate object detection and localisation. Given a set of features belonging to an object in the source frame, F_o, the reverse transform (translation, rotation and scale changes) from an arbitrary point in the image (for example, the centre of the object, $\overline{F_o(\mathbf{x})}$) to each of the features in F_o is calculated. This is the "star", with each point being the reverse transformation from the arbitrary point back to the location of the feature it was calculated from. To search for the object in the frame, features in the target frame are first tentatively matched to the object features by their descriptors in the standard way. For each correct match the "star" is warped based on the canonical scale and rotation of the feature, centred around the feature location. The locations of the resulting transformed star points of the star "vote" for the likely location of the arbitrary point in the target image, as shown in Figure 2.12 (top, right). By placing the star pattern of the original image onto the target image, a match likelihood based on distance between the projected features F_o (given by the star pattern) and the target features can be used to identify better matches. A worked example is shown in Figure 2.12.

Recall the use of the Hungarian matching algorithm to propose feature matching as an energy minimisation problem. Disregarding for a moment the problem of high computational cost, another problem is that the standard linear assignment problem does not allow for pair-wise interactions between matches, i.e. the additional cost of having a match that disagrees with other matches in the vicinity. For example, the disparity given by a match between two features should be similar to the disparities given by matches between nearby features.

Torresani et al. [173] propose an impressive energy minimisation framework that encourages local smoothness in feature disparities, and the descriptor comparison, feature neighbourhoods and match rejection are taken into account in the same system. Using a measure from shape matching literature [18], a cost is calculated for how well a given match between feature f_p and f_q agrees with the matches between those in the neighbourhood sets N_p and N_q. In shape matching, the neighbours of feature points are implicitly defined by the contour of the image; each feature point has exactly 2 neighbours. Torresani et al. defines the feature neighbourhood set, N_p, of a feature point as simply the set of features within a fixed radius of f_p. A demonstration of improvement between the two energy minimisation schemes when applying spatial consistency is shown in Figure 2.13.

The work of Torresani et al. presents a logical, intuitive framework for encouraging neighbourhood match constraints. This represents the state of the art in feature matching. Ignoring the extremely high computational cost in finding the global minimum match energy for a moment, it can be seen in Figure 2.13 that there are still image regions lacking any matches. In

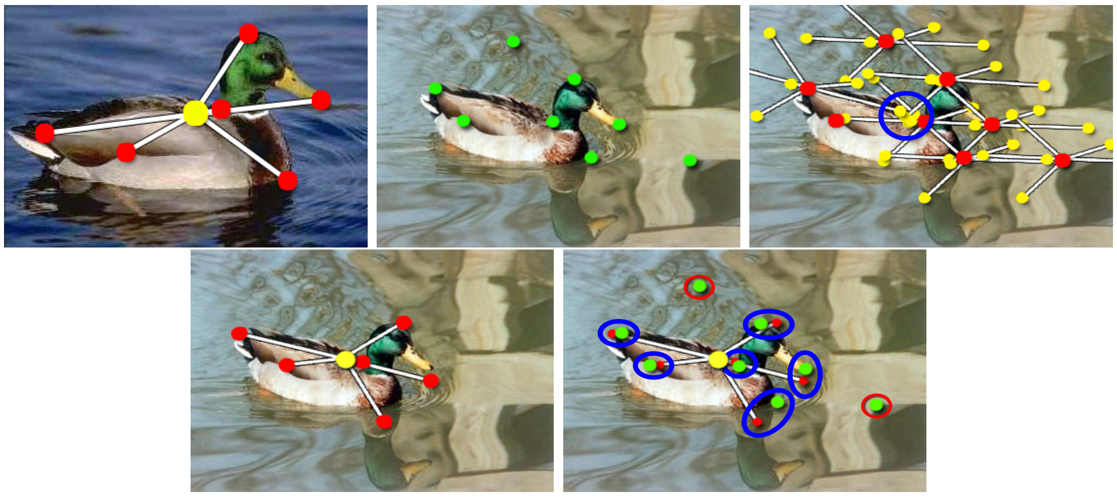


Figure 2.12: Using the star-graph model to improve feature matches. Query feature points F_p (red) are found in the user selection, and the polar translation (ρ, θ) from each point to the centre of the duck c (yellow) is calculated, top left. Note that c can be an arbitrary location in the image, in most cases it's convenient to place it relative to an object of interest, i.e. the centre of the duck, found by taking the mean location of the detected features (red). Features in another frame F_q (green) are calculated and matched against features F_p, top, centre. Note the two incorrect matches. The polar translations of F_p are then applied to each of the features in F_q, projecting the candidate centres of the object c relative to F_q into the new frame (yellow), top, right. The most likely location of c is given by c', shown by large clusters of points (blue circle). At the site c', features F_p are mapped into the new image as F'_p using the reverse of (ρ, θ), bottom left. Finally, the distances between F'_p and F_q can be used as a spatial likelihood to help identify correct (green) and incorrect (red) feature matches, bottom right.

difficult matching situations, there are often regions that simply do not have enough reliable information (either appearance or spatial) to obtain reliable matches. In cases where image correspondences in these regions are required, it is necessary for the user to supply additional information. Part of the work of this thesis looks at using the framework of Torresani et al. to intelligently incorporate user information for better matching.

2.5.2 Improving Nearest-Neighbour Descriptor Matching

Most spatial consistency schemes often rely on an initial stage of establishing putative matches between the image, where it is assumed a high proportion of correct matches will be obtained, often by nearest neighbour (N.N.) descriptor matching. For a typical PAL sized image, a typical detector using default parameters may detect between 2,000 and 10,000 features. Using an exhaustive nearest neighbour search on the descriptor space is computationally expensive, in some cases taking longer to match than to detect and describe the features.

Assume that the chosen detector and descriptor system produce points that when matched

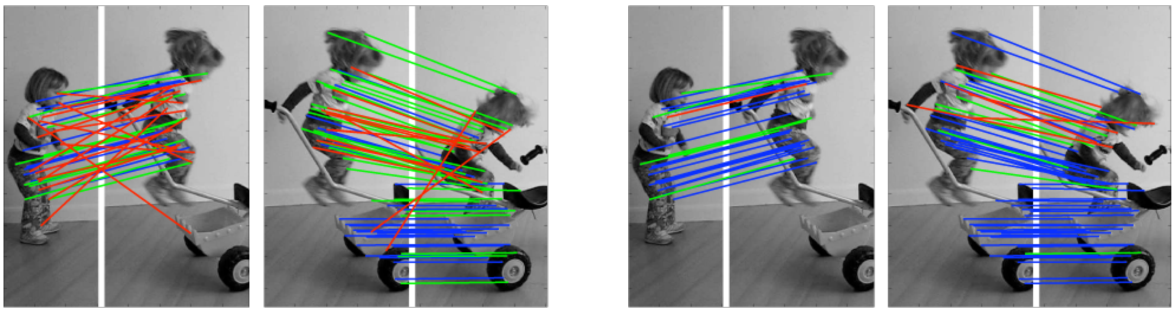


Figure 2.13: Example of matching features using energy minimisation Hungarian matching (linear assignment) (left), and the spatial energy minimisation scheme of Torresani et al. (right). The blue lines show correct matches, while the green and red lines show missed assignments and incorrect matches respectively. Notice the significant increase in correct matches when using the spatial consistency framework of Torresani et al. (Images reproduced from Torresani et al. [173].)

using nearest neighbour search with a certain descriptor distance threshold, result in matches with a certain false alarm rate. If the threshold is lowered, fewer matches will be accepted, and the false alarm rate is lowered also. This is the standard way to reduce Type 1 error. A simple way to detect more correct matches is to increase the number of candidate matches (by adjusting detector parameters) and lowering the threshold accordingly, such that the number of correct matches remains approximately constant, with a lower false alarm rate. In practice, the threshold and false alarm values rarely share a linear relationship, but the idea usually holds; more candidate matches allows for more correct matches[2]. Of course, this means that the matching task grows dramatically; for an N fold increase in features per image, the number of candidate matches increases N^2 fold. A fast matching algorithm is therefore of great importance to a feature-based application.

Beis & Lowe [15, 105] propose an interesting method of looking up high dimensional query points in k-d trees, which they call the *best bin first* (BBF) algorithm. Consider N points with k dimensions. The objective of k-d trees is to generate a balanced binary tree by iteratively partitioning the points at each level into two approximately equal sized sets. In the BBF algorithm, this partitioning is achieved by finding a threshold m, in the dimension i exhibiting the highest variance. The threshold m is given by the median value of dimension i of the point data at the current level. To perform a N.N. lookup for a point q, the closest bin is found by first traversing the tree to give a good approximate nearest match. Nearby bins are then searched to find the actual nearest neighbour, as the dimension partitioning scheme does not necessarily mean q will be "close" to its quantised bin in L^2 or χ^2 space for example. The problem with performing N.N. matching using k-d trees in high dimensional spaces is that the number of nearby bins to search for the nearest can be very large. To improve the speed of matching, the

[2]The caveat is that at some point, the number of candidate matches becomes too high, and the cost of simply performing dense per pixel matching is lower than detecting and matching features.

BBF algorithm orders the bins to be searched in order of increasing distance from the query point q, and also places an upper bound on the number of bins to be searched. This allows the tree search to be terminated sooner with a high probability of being close or at the nearest neighbour. Experiments by Lowe showed that for a database of 100,000 feature points, the speed of approximate nearest neighbour search was increased by nearly 2 orders of magnitude, with less than a 5% loss in the number of correct matches.

Another way of looking at the problem is to reduce the computational cost of comparing a pair of descriptors. An obvious way to speed up matching is to reduce the number of elements in the descriptor. However, simply removing elements decreases the the ability of the descriptor to discern between image patches. A more intelligent way of reducing the descriptor size is needed. Ke & Sukthankar propose using principal component analysis (PCA) to reduce the dimensionality of the feature descriptors before matching [81]. In their example, the SIFT descriptor is examined. Using PCA makes sense as there is likely to be some redundant information included by the way the descriptor is calculated. For example, the way the gradients are indexed into the histogram bins in the SIFT descriptor means that there will be some correlation between neighbouring spatial bins in the feature vector. The idea is to first collect a large number of feature descriptors, preferably from (or representative of) the image corpus the matching system will be used on. PCA is then applied to the set of descriptors to extract the descriptor variance basis (eigen-) vectors and their corresponding variance contributions (eigenvalues). By removing basis vectors corresponding to low variance (i.e. where there is little information), a transform matrix can be applied to a new descriptor to reduce the feature vector dimensionality. Ke & Sukthankar [81] show that the 128 element SIFT descriptor can be reduced to 96 elements with minimal loss in matching accuracy, indicating a large amount of redundancy in the original descriptor, an example of the redundancy is shown by a sample covariance matrix in Figure 2.14. Instead of throwing data away, a different transform matrix can be designed by re-arranging the basis vectors in order of decreasing variance. By applying this matrix to a descriptor, a new descriptor is calculated with elements arranged according to variance. Now when performing an exhaustive nearest neighbour search, candidate descriptors can be rejected earlier (i.e. after the first couple of elements) resulting in a significant speed increase with no loss in precision.

Previous sections focused on comparing features found in one image to those found in another image with similar object structure, i.e. a one-to-one matching relationship. For an object recognition application, comparing a query image against every stored instance image of the object in the database using the previously described correspondence matching is infeasible for a database of any useful size. In the following section, the many-to-one relationship is discussed.

2.5.3 Database Matching

During typical database matching applications, such as object classification or recognition scenarios, a query image is presented to the system and is asked to detect or classify the object

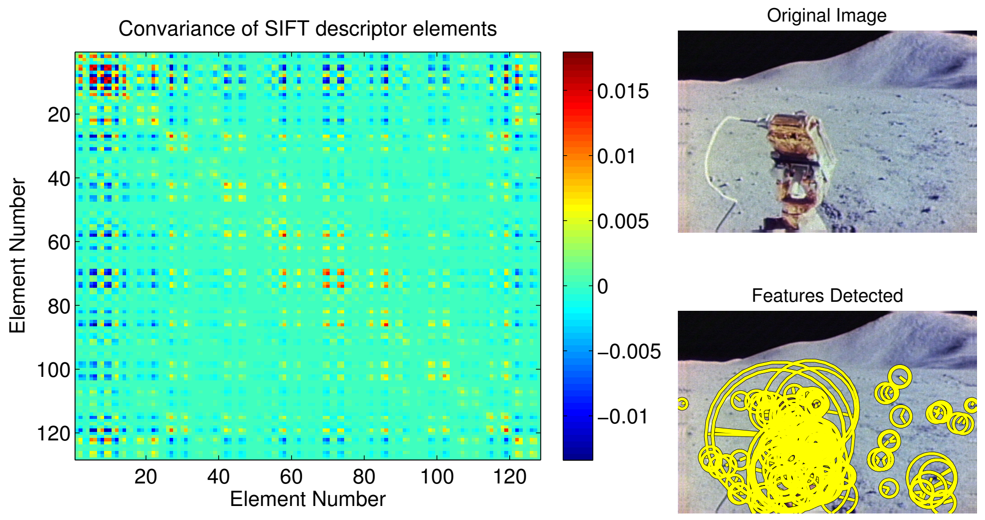


Figure 2.14: Covariance matrix (left) between elements of approx 100,000 SIFT descriptor calculated on features from NASA Lunar explorer video footage (right). Repeating patterns of negative (high magnitude) covariance (blue, around the border), spaced evenly approx. 8 elements apart, with higher magnitude (darker blue) ones spaced 32 apart, corresponding to the neighbouring regions of the 4×4 SIFT descriptor spatial bins. The covariance can be exploited to reduce the number of descriptor elements.

in the scene. Instead of identifying correspondences between the features of the query object and those of every object instance in the database, the key is to generalise the features detected in an image, allowing the image content to be represented (and compared) in a more compact form. For example, by quantising the descriptors using a dictionary of feature vectors, an image can be represented by a list of codebook entries at various spatial locations.

Sivic & Zisserman [162, 160] exploit research from Information Retrieval (IR) literature to cast the traditional object detection problem into a query-by-content database search problem typically used in text-based search engines. Their system allows the user to search for an

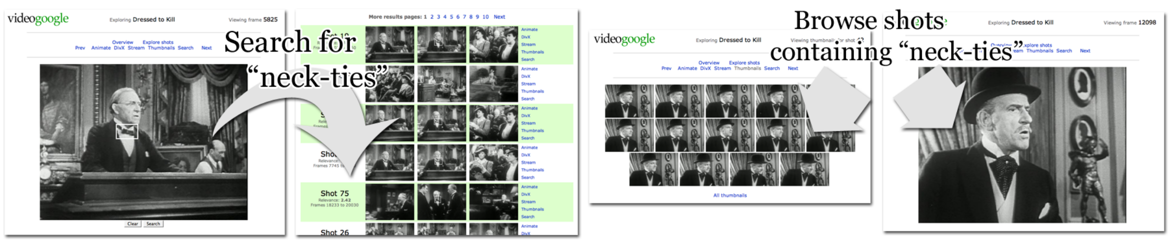


Figure 2.15: Examples of query-by-content object searching through video by Sivic & Zisserman [161, 160]. By treating detected features as "visual words", common text-based information retrieval techniques can be applied to video, giving real-time interactive object detection in large databases.

object throughout a long ($>$ 2hr.) video in real-time, using an example image of an object (typically one already in the video) as input, as shown in Figure 2.15. The video to be searched is pre-processed, where features calculated on key-frames (shot boundaries, or every 10-15th frame), and their descriptors are (k-means) clustered to find a vocabulary of "visual words". The number of words in the vocabulary is typically around $k = 10,000$. The feature descriptors throughout the sequence are then quantised, with each feature now described by the index of the nearest cluster centroid. A video key-frame can now be represented by the list of quantised feature point indices contained in the scene. This is known as the "bag-of-words" model in IR literature, where a "document" (video-frame or object) is represented only by the contained "words" (feature indices), the ordering of which is disregarded.

Nister & Stewenius [125] extend the work of Sivic & Zisserman by building a hierarchical quantisation framework, allowing faster look-up, and dictionaries with far higher numbers of visual words (16M, as opposed to 10,000 used by Sivic & Zisserman) resulting in better feature discrimination. The tree structure is generated by hierarchical k-means clustering with a fixed branching factor. For example, if the branching factor, $K = 10$, the first level of the tree has 10 nodes. Each node then branches to 10 other nodes and so on, resulting in K^n leaf nodes, where n is the number of levels. When the system is presented with a feature to be quantised, the descriptor is first compared to the 10 descriptors belonging to the nodes, given by the cluster centroids. The nearest neighbour to each of the 10 nodes is found, and the corresponding branch is followed to the set of 10 nodes at the next level. This is repeated for all remaining levels. The codebook index for the feature is given by the path to the matched leaf node, i.e. if $n = 6$ levels, the codebook index is a 6 digit base-K number defining the traversed path down the tree. The power of the quantised descriptor in the "bag-of-words" model becomes apparent when the task of object detection and localisation in a massive image database becomes as trivial as a simple text-based search, as follows.

Searching for an object within the video is the equivalent of comparing "documents", where the "query" document is the set of feature indices belonging to the object, and the documents to be compared against the query are the sets of feature indices in each of the key-frames. A popular similarity measure in IR literature is the Vector Space Model (VSM) [59]. Consider representing a document by a column-vector, $\mathbf{v}$, the number of rows of which is the number of visual words in the dictionary, N_c. The entries in $\mathbf{v}$ are the frequencies of the visual words in the document. The number of dictionary words, N_c, comes from the data-reduction technique, i.e. k-means clustering (where $N_c = k$), or the hierarchical clustering of Nister & Stewenius (where $N_c = K^n$). The similarity between two documents, $\mathbf{v}_1$ and $\mathbf{v}_2$, is given by the cosine of the angle between the two, i.e. $\cos\theta = \frac{\mathbf{v}_1^T \mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|}$. Intuitively, this measures the relative differences in frequencies.

For example, consider a contrived text example of two documents, where the second document is the concatenation of the first document with itself. As the second document contains the same vocabulary of the words, with twice the frequencies, the vectors representing the two documents would look like $\mathbf{v}_1$ and $\mathbf{v}_2 = 2\mathbf{v}_1$. Comparing the two documents with the L^2 distance results in value of $\sqrt{\mathbf{v}_1^T\mathbf{v}_1}$. However, the cosine of the angle difference is 1, indicating an angular difference of 0 between the two vectors. From an information retrieval point of view, the content of $\mathbf{v}_1$ and $\mathbf{v}_2$ is the same, therefore the cosine difference is more useful in measuring the semantic content of the documents than the L^2 distance[3].

An interesting problem arises when using the VSM; the model assumes the words have equal importance, for example, the semantically irrelevant word "the" has the same influence on the similarity measure as the word "apple". This concept applies to visual words too, some image features may be too generic, and not as useful as others or describing an object. Using the data in the corpus, the relevance of words can be measured by their relative frequencies in the corpus. One weighting method popular in both text- and vision-based literature is the *term-frequency inverse document frequency* (tf-idf) [77].

The tf-idf weights descriptors according to their descriptive power, and is composed of two parts. The *term frequency* raises the importance (weighting) of a word the higher the occurrence of the word in the document, as it is expected that this word is useful in describing the current document it is contained in. While the inverse document frequency lowers the weighting of the word if it occurs frequently throughout the corpus. For example, a document containing many occurrences of the word "apple" is probably related to apples. If the text corpus being searched is a collection of documents about fruit then "apple" is a highly descriptive word. However, if the corpus is a library of books specifically about apples, then the word "apple" is probably not as descriptive, and will be weighted lower. Returning to images, consider an image database with features clustered to a dictionary of N_c visual words. Given N_f corpus images in the database, the weight t_{if} for every code-book entry $i \in \{1, \ldots, N_c\}$ in all images $f \in \{1, \ldots, N_f\}$ is then calculated

$$t_{if} = \frac{n_{if}}{n_f} \log\left(\frac{N_f}{n_i}\right)$$

where n_{if} is the number of occurrences of the code-book entry i in image f, n_f is the number of features in f, and n_i is the number of images in which code-book entry i occurs throughout the corpus. Intuitively, the feature weight t_{if} balances descriptive power between very uncommon and very common features in two ways: the fractional term, $\frac{n_{if}}{n_f}$, weights a feature i higher if it occurs frequently throughout the image f, while the log term, $\log\left(\frac{N_f}{n_i}\right)$, penalises feature i if it occurs often throughout the image corpus. Together, a feature will have a high weight if it is popular within the image, yet does not occur frequently throughout the database. To compare a pair of images, again using the cosine of the vector angle difference, the previous vectors $\mathbf{v}_1$

[3]In this example, the normalised L^2 distance, i.e. $\|\frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} - \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|}\|$ would give a distance of 0, indicating strong similarity. However, consider the trivial example where $\mathbf{v}_1 = [0, 1]^T$ and $\mathbf{v}_2 = [1, 0]^T$ i.e. a dictionary of size two, where the documents $\mathbf{v}_1$ and $\mathbf{v}_2$ contains one instance of each word. Despite both documents containing no semantically similar content, both the regular and normalised L^2 distances will give a distance of $\sqrt{2}$, while the cosine distance will be 0. Again, the angular difference of the normalised vectors captures better the relative difference in semantic content.

and $\mathbf{v}_2$ are simply replaced by $\mathbf{t}_1$ and $\mathbf{t}_2$, where $\mathbf{t}_f = \{t_{1,f}, \ldots, t_{N_c,f}\}$.

The tf-idf weighting scheme is particularly important in "visual word" feature applications for a number of reasons. One such reason is to remove the bias a feature detector has in detecting certain regions in its associated low-level feature space. For example, if the distribution of corner shapes detected by the Harris corner detector is analysed for a large set of random images, a higher proportion of orthogonal corners will be observed, simply because corners at 90° give larger corner responses. From a semantic point of view, right-angled corners are not any more or less meaningful than acute- or obtuse-angled corners. This preference for right-angled corners can be viewed as an artefact of the detector, and is compensated for by the weighting scheme. A more important reason for employing the tf-idf weighting is to focus the context of the detection or classification system to the images present in the database, identifying the relevance of each feature in an image relative to the corpus.

In later work in this thesis, VSM's are used to help reduce false alarms in parsing sports coaching video, and also in an object detection-like scenario to diagnose when user input is required to aid automatic object segmentation in video. In both these applications, the image content throughout the videos is extremely similar. For example, in the sports coaching video the camera is generally fixed on the same location, with the same athlete in shot for the duration of the coaching session. In the object segmentation case, the videos being segmented are contiguous shots, usually containing the same semantic content throughout. In both applications, it is expected that many of the same features (for both object and background) will exist throughout the sequence. The limited number of distinct features, combined with the persistence of uninformative ones, makes it difficult to distinguish the object in the frame using the VSM alone. It will be shown that by calculating the tf-idf for each video, the context of the object detection system is focused; features most relevant to this video (and therefore in detecting objects within the video) are weighted higher.

2.5.4 Comparing Feature Detector Performance

In any practical situation, the user wants to use the best detector possible for the application. As discussed previously, there are a number of factors that make it difficult to determine what detector is most suitable. For example, whether to manually create ground truth on a number of application images, or to simply use the results of a detector or descriptor comparative survey. To begin this discussion, two application specific methods of feature comparison are now presented.

Moreels & Perona [121] present an interesting approach to evaluating descriptor performance. Realising that manually creating ground truth is difficult, and in many cases unreliable, Moreels & Perona propose an automatic method of simultaneously generating ground truth and evaluating the descriptor. The idea is that correct matches are found by verifying proposed matches against the epipolar geometry constraints between sets of calibrated images (from multiple views of the same scene). This not only measures the performance of the detector or descriptor or matching algorithm, but rather the entire system as a whole. The disadvantage is that this evaluation procedure is only relevant for multi-view geometry applications, where the transforms between cameras can be found with a high accuracy. From a practical point of view, evaluating the system in its entirety is very appealing, and forms part of the motivation for later work in this thesis.

Nowak & Triggs [127] present an interesting investigation into the effects of image sampling on object classification performance in the "bag-of-features" model. Instead of addressing the problem of identifying the "best" feature detection system for the task of object classification, Nowak & Triggs take a deeper look at a more fundamental level of how features should be selected to best describe an image. The authors conclude that the number of interest points is the most critical factor in a successful system. This becomes clear when the extreme case of oversampling is considered; if every pixel is sampled as a feature point and represented by a typical SIFT descriptor, the likelihood of missing a match when one exists (Type 2 error) is minimal. Nowak & Triggs posit that sparse feature detectors do not detect enough regions to give equivalent classification accuracy as oversampling. In their experiments, a feature detector that samples random locations with various numbers of samples is shown to provide reasonably high accuracy. However, the authors acknowledge that the choice of detected locations in sparse non-random detectors is important, and that for a random detector to achieve an equivalent classification accuracy, the number of sampled points is much greater. Mikolajczyk et al. [117] also discuss the importance of feature sampling, noting that some of their results should not be generalised, as they may be "statistically unreliable for much larger numbers of features".

Looking at application agnostic comparative studies instead, Mikolajczyk et al. [117] present the most comprehensive study into the performance of feature detectors. The authors propose to measure the *repeatability* of a given detector to a variety of image conditions, such as blur, image noise, compression artefacts, changes in scale, rotation and translation. Sets of images are grouped according to the condition to be tested. Repeatability in this case is defined as the average percentage of correct matches over all images sets. A ground truth is created by manually providing correspondences between images of the set, and estimating an approximate homographic transform from the matches. Using the approximate homography as a starting point, an automatic process refines the homography using the method of Hartley & Zisserman [71]. Feature points are calculated on each image in the set using the candidate detector. One image in the set is designated the reference image. For every non-reference image in the set, the homography between the reference and non-reference image is used to project points from the first image onto their expected locations in the second image. For each candidate point correspondence given by the projected points, the overlap between the support regions (the pixel regions typically used for calculating descriptors) of the feature point pair is measured, and the match is accepted if it is within a threshold. The various feature detectors are then ranked in order of performance. Similar comparative studies exist for evaluating descriptor performance [116, 183].

The application agnostic comparative evaluation studies not only give a good overall impression of detector and descriptor performance, but also a protocol for how to manually extract ground truth for a range of tests and use it to derive meaningful comparisons. The application specific comparisons using automatically extracted ground truth are interesting as they present both a comparison, and a repeatable protocol for comparison experiments. Both approaches require the use of ground truth to estimate performance, however in many cases it may not be possible to automatically extract ground truth, or may be too laborious to manually calculate in the range-of-tests case. This difficulty in acquiring and using reliable ground truth motivated development of an application specific protocol that uses a very simple form user supplied ground truth as a basis for comparison; the actual images used in the application. This work is presented later on in the thesis.

2.6 Summary

In this chapter, the reader has been introduced to some of the more interesting feature based applications, and how local features can be used to infer meaningful high level information. A discussion of the difficulties facing practical feature matching systems was presented, followed by the state of the art in feature detection, matching features, and evaluation of feature based systems. In particular, two areas for further investigation have been identified; improving feature correspondence matching, and how to effectively evaluate feature detector systems.

The remainder of this thesis focuses on three aspects of local features. Firstly, applying aspects in the state of the art of local feature techniques to applications in areas usually unrelated to features, such as video object segmentation and sports video parsing. Secondly, developing a protocol that allows for subjective comparisons between feature detection and description systems for application specific measuring of feature detector performance. Finally, how to encourage feature correspondences in difficult regions by incorporating user information.

Chapter 3

A Protocol for Application Specific Feature-Detector Comparison

The choice of detector and associated parameters is critical to the overall performance of an application involving feature points. However, choosing the most suitable detector is tricky, as ground truth is difficult or often impossible to establish, and the user is left to trial-and-error. This chapter presents a simple method to compare performance between different detectors, which is specific to the user's application. At some stage, most feature point applications rely on exploiting appearance similarities between the image data at detected feature locations, for example, *descriptor* comparison. A measure of detector performance is derived by analysing distributions of descriptor distances between matched features. By estimating the distributions from the user's application image database, this protocol results in a more relevant detector comparison. Experimental validation of the protocol is presented, followed by example applications of video tracking, object and face detection for a number of popular detectors.

3.1 Introduction

The success of an application using feature points is largely dependent on the performance of the chosen feature detector. However, selecting the best detector for the application is not simple. The work presented in this chapter proposes a black-box method of evaluating many detectors and parameters, where the only input is the user's own target application image database.

There have been many comprehensive studies evaluating and comparing various feature detectors and image region descriptors [116, 117, 188, 121, 113, 153, 183, 166], as discussed earlier in Chapter 2. The work of Mikolajczyk et al. [116, 117] is the first to attempt an exhaustive feature comparative study, and conclude that measuring performance independent of a given application is difficult. Mikolajczyk et al. use manually created homographies to estimate the number of correct matches and localisation accuracy for a variety of detectors. The work of Moreels et al. [121] perform similar experiments, using epi-polar ground truth data from calibrated cameras instead of manually fitting transforms. Although comparative studies such as these provide insight into the choice of feature detector, the ultimate effectiveness of a feature detector and parameters depends largely on the application image data. The works of Seeman et al. [153] and Zhang et al. [188] use classification results for specific applications to measure descriptor performance, and Stoettinger et al. [166] uses both the evaluation criteria of Mikolajczyk et al. and an image retrieval example application to measure performance. Naturally, application-based tests are more relevant to users looking to replicate the presented applications exactly. However, these studies do not necessarily represent expected performance for other applications. In addition, differences between detector implementations or choices of parameters used in these studies can greatly affect performance results. Given that during application development, an image database of example content specific to the user's own application is usually available, it makes sense to use this image data directly to somehow evaluate candidate detectors and parameters. Presented in this work is an image centred approach to evaluating feature detectors and parameters through analysis of the distributions of matched feature descriptors, relevant specifically to the target application.

Of course, in many real-world scenarios, the choice of feature detector is often decided by practical considerations, such as the availability of detector source code, computational efficiency, or licensing issues. In such situations, a detector satisfying these criteria is incorporated into the application, and the performance of the overall system is evaluated at the end of development. To try to improve performance, various permutations of other feature detectors, descriptors and associated parameters are usually swapped in and out of the system, and evaluated in a trial and error manner. This "swap in-and-out" evaluation method is laborious and makes it difficult to isolate the effects of changes in the feature detection stage on overall system performance. For example, machine learning algorithms can compensate for some problems in feature detection. This work presents a method of evaluating the feature detection stage independently of the rest of the system.

Section 3.2 describes the proposed method of feature point evaluation, and provides validation of the method against ground truth. Section 3.3 demonstrates how feature detector performance is measured for example applications, while Section 3.4 provides a conclusion and discussion.

3.2 Feature Comparison Protocol

At some stage, most applications using feature points involve comparing descriptor vectors [116, 121, 183]. For example, object classification systems are trained on sets of descriptors from annotated exemplar images, video tracking and image registration systems use the descriptor vectors to establish initial putative matches before applying spatial consistency constraints, and content-based image retrieval (CBIR) systems quantise descriptor vectors for indexing and retrieval. Descriptors are useful because the calculated distance between a pair of descriptors (for

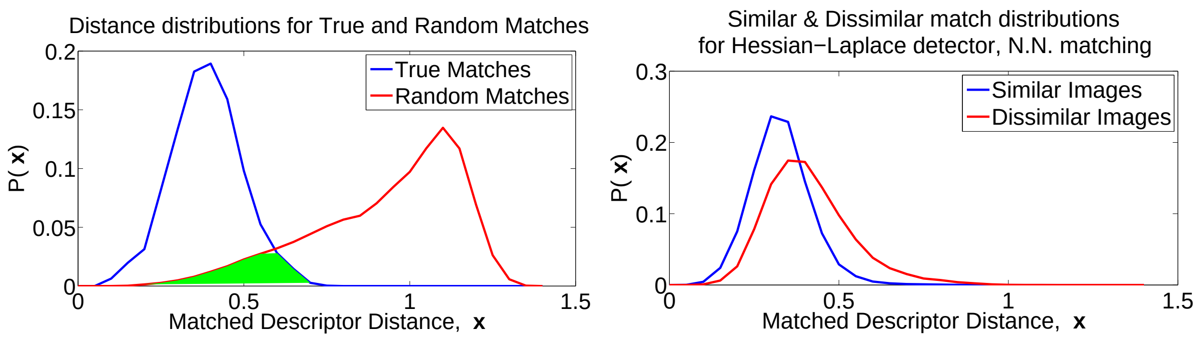


Figure 3.1: Left, distributions of feature descriptor distances for true (blue) and known to be incorrect, random matches (red) identified using ground truth data discussed in Section 3.2.3. The descriptor distance $\mathbf{x}$ is given by the Euclidean distance between two descriptors, i.e. $\|\mathbf{d}_1 - \mathbf{d}_2\|$. The overlap in distributions makes classifying matches correctly difficult, separating these distributions improves match reliability. Right, distance distributions of matches identified by their nearest-neighbour *(N.N.)* in descriptor space, between similar (blue) and dissimilar (red) images using the Hessian-Laplace detector [114]. It is proposed that the separation between the estimated distributions of similar and dissimilar application specific images is related the amount of true and random distribution overlap (left, green), and can be exploited as a useful measure of detector performance.

example Euclidean or χ^2) gives a measure of how similar the two images patches being compared are. The presented work uses the SIFT descriptor [105] to summarise image regions, and the Euclidean distance to measure descriptor distances. Other distance functions were evaluated, but no significant differences between results were observed.

3.2.1 Characterisation of Detector Performance Based on Distance Distributions

Consider matching features using descriptors between a pair of ground truth images for which the mapping between every point from one image to the other is known. Now consider plotting two distributions of the distances between matched feature descriptors, one for matches that can be identified as correct, and the other for those known to be incorrect. By doing this, a feeling is obtained for what values of descriptor distances for correct and random matches should look like for real world images. An example using ground truth images (discussed later in Section 3.2.3) is shown in Figure 3.1 (left), where the descriptor distance distribution of correct feature matches is shown in blue, and the descriptor distance distribution of matches matched intentionally to incorrect, random image regions is shown in red. For the random match distribution, matches which were found to be correct by chance were rejected. The number of samples for the correct and random match distributions were $8,632$ and $470,666$ respectively, the random distribution naturally has more samples as it is easier to find bad matches that it is to find good ones.

As expected from looking at Figure 3.1 (left), the correct matches have lower matched descriptor distances than incorrect matches. However, notice the overlap (green) between the two distributions. It is this overlap that causes problems in correctly identifying matches, i.e. incorrectly accepting or rejecting matches. As noticed by Winder et al. [183], matching is improved if this overlap is decreased. This is effectively the same as increasing the separation between the two distributions. Therefore, a good feature detector should select features such that when descriptors are compared (and used to identify matches), a wide separation between correct and incorrect matches is observed. The author posits that if the the separation between distributions is measured, the relative performance of a feature detector can be estimated.

In real world scenarios the distribution of true matches is not known, however the amount of overlap can be approximated. In this work, it is assumed that features matched to their *nearest-neighbour* (N.N.) in the descriptor space results in correct matches. This assumption is reasonable for very similar images and becomes less valid as the dissimilarity between images increases. Consider calculating features on a pair of images with similar content, and matching their descriptors by finding the nearest-neighbour. Now consider matching descriptors between a pair of dissimilar images using nearest-neighbour, and plotting the distributions of the descriptor distances for the similar and dissimilar image pairs. An example is shown in Figure 3.1 (right). The distributions are closer together than those of Fig. 3.1 (left), highlighting the difficulty in correctly classifying possible matches using nearest-neighbour. It is exactly this matching difficultly that we want to measure. Given that the distribution belonging to the dissimilar matches is known to be from random (N.N.) matches, the separation between these distributions is related to the separation (and therefore overlap) of the true and false match distributions. The example in Figure 3.1 shows the distribution of match distances for one image pair only. It is proposed that by calculating and comparing the similar and dissimilar distributions from *many* images of the target application, a generalisation of the distribution separation can be found for a given detector, yielding a useful measure of detector performance.

The task now is to measure the separation between the pair of distributions. There are various ways of comparing two distributions, popular examples include ANOVA, and paired t- or z-tests. These measures make implicit assumptions about the underlying data. The shapes of the distributions of distances between descriptor vectors are not easy to define, and so a non-parametric distance measure is preferred. Winder et al. [183] calculate the integrals of the correct and incorrect distributions, and use them to estimate an ROC curve. The ROC curve plots the correctly detected matches as a fraction of all true matches against incorrectly detected matches as a fraction of all true non-matches. The area under the ROC curve is a measure of the distribution overlap, and hence detector / descriptor performance. This is a sensible metric providing detailed ground truth can be extracted and is available. This is not assumed for the scenario of the proposed comparison protocol. Instead, this work uses a similar non-parametric measure to estimate the area of overlap, the Kolmogorov-Smirnov (K.S.) distribution comparison. The K.S. test statistic K is the maximum divergence between the empirical cumulative distribution functions (c.d.f.'s) of the two distributions to be tested, i.e.

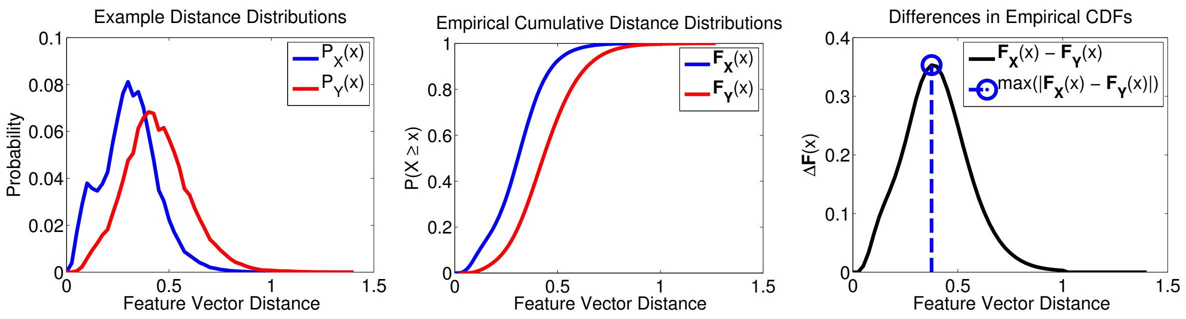


Figure 3.2: Example Kolmogorov-Smirnov distance. Typical distributions of descriptor distances between matched points from similar (blue) and dissimilar (red) images are shown. The test statistic, K, given by the maximum divergence in c.d.f.s, (centre & right), captures the overall "shift in mass" of the distributions. In this example, the K.S. measure is $K = 0.36$.

$K = \max(|\mathbf{F_X} - \mathbf{F_Y}|)$. An example is shown in Figure 3.2, with additional details provided in Appendix C. This effectively measures the maximum "shift in mass" without any prior knowledge of the shapes of the distributions.

3.2.2 Measuring Detector Performance against an Application Image Database

The process of generating the distributions from application image databases is presented. To put this idea into context, a typical object recognition application is presented which is extended later to other applications. For this example, let the set, Φ, contain all of the detector / parameter combinations of interest to the user to be investigated for classification performance. For example, an element of the set Φ could be the SIFT feature detector with a DoG threshold of 0.1, while another element could be the Harris-Laplace detector with a corner threshold of 0.001. The objective is to find the detector and parameter combination, $\phi \in \Phi$, that gives the highest separation in distributions of descriptor distances for matched features of images for similar (intra-class) and dissimilar (inter-class) object classes. The sets of all intra- and inter-class image pairs, denoted A and B, are defined as follows.

Imagine an image database D to be used in the classification system with M classes. Sets of images representative of the particular object class m are given by $D_m \subset D$, where $m \in \{1, \ldots, M\}$. When comparing intra-class (similar) images for object class m, every image in D_m is feature matched to every other image in D_m. The set A_m of intra-class image pairs to be compared for class m is given by $A_m = [D_m \times D_m]$, with the set of similar image pairs over all classes given by $A = \{A_1, \ldots, A_m\}$. When comparing inter-class (dissimilar) images, images from a set m are compared to images from other image class set n, where $n \in \big\{\{1, \ldots, M\} \setminus m\big\}$. The set B_m of inter-class image pairs to be compared between sets m and n is given by $B_m = [D_m \times D_n]$, with the set of dissimilar image pairs over all classes given by $B = \big\{B_m \;\forall\; m, n \in \{1, \ldots, M\}\big\}$. The elements of the sets A and B are image pairs.

For each class m and detector combination, $\phi \in \Phi$, local features are detected on all the image pairs in the set of similar images A_m, and the image pairs in the dissimilar image set B_m. Correspondences between each image pair are then given by N.N. matching, and the descriptor distances for all the matches in sets A_m and B_m are collected for analysis. From the match descriptor distance distributions (of A_m and B_m), the separation K_m between is then measured using the K.S. test statistic to rate the performance for the combination ϕ. In this way, the impact of any detector combination ϕ for the class m is reduced to the single number K_m. The mean and variance of the distribution separations, μ_K and σ_K^2, provide a meaningful representation of the performance of the detector ϕ across the entire application image database D. An ideal detector would have a μ_K of 1 (the maximum value of the K.S. test, and so also implying a value of 0 for σ_K^2), indicating that there is no overlap between similar and dissimilar match distributions for all classes.

This method of measuring performance extends to any other feature based application by changing how the sets of image pairs A and B are built. For example, in a video tracking application, the set of similar image pairs A could encapsulate pairs of consecutive frames, where it is expected that the image content between the two frames is similar. Then, pairs of frames known to contain different content could comprise the set of dissimilar image pairs B. Further examples are presented and discussed in Section 3.3.

3.2.3 Experimental Validation

The proposed measure of performance is now compared to a ground truth to validate its use as a measure of detector performance. For this work, ground truth was generated for a set of wide baseline image pairs (45° or 90° apart) using the multi-view space carving work of Starck et al. [164]. The idea is that space carving can generate very accurate, dense models of 3D objects given a large amount of views (8 in this case), and the correspondences between pairs of arbitrary locations can then be generated from the estimated 3D meshes.

The image database used for the validation contains 19 still scenes, each from 8 angles. For each scene m, two pairs of camera angles 45° apart and two pairs of wider camera angles 90° apart are compared, resulting in 76 image pairs in total in the similar images set $A = \{A_1, \ldots, A_{19}\}$ to be compared. Examples of the multi-view scenes are shown in Figure 3.3. The set of dissimilar images $B = \{B_1, \ldots, B_{19}\}$ is comprised of 76 random image pairs from the database, ensuring that the image angles are greater than 90°, and the pair are not from the same scene. For each scene m, feature correspondences are found using each of the various feature detectors and parameter settings Φ (presented below), between the image pairs in A_m, and then in B_m. The separations between the distance distributions for the matched features for A_m and B_m are calculated. Using the ground truth, the percentages of correctly matched features per image pair is determined. Results for the “Character 1” and “Fashion 2” scenes, at 45° and 90° camera angles over a range of detector combinations are shown in Figure 3.4.

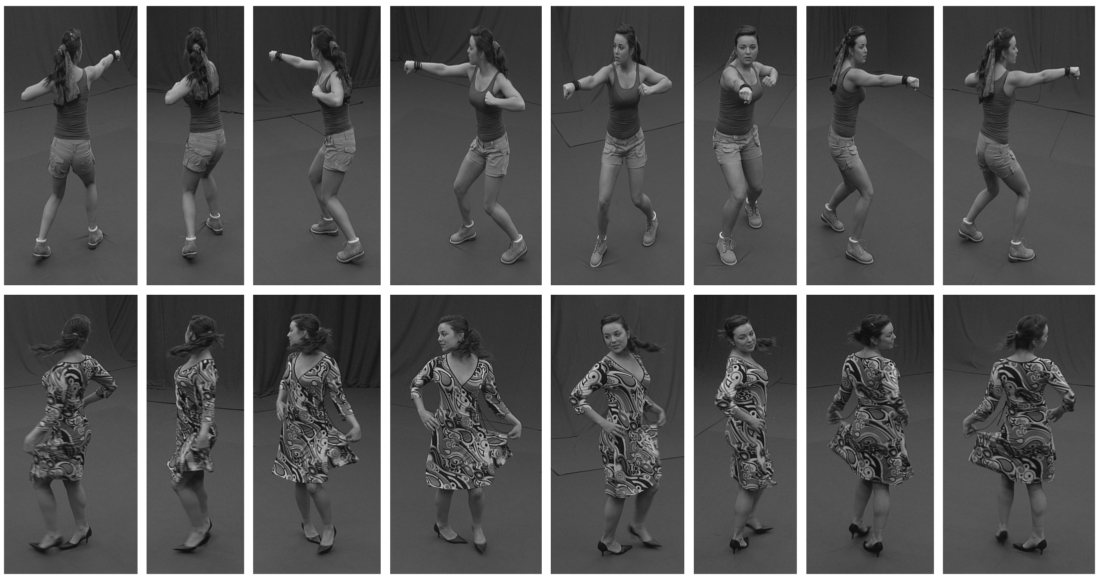


Figure 3.3: Example multi-view images from eight cameras of the “Character 1” and “Fashion Twirl” scenes (top and bottom). From left to right, the camera angle in each consecutive figure is 45° apart, captured simultaneously.

Figure 3.4 shows that generally, with increased K values the percentage of correct matches also increases. The results for other scenes (not shown) exhibit similar relationships. In general, the correlation values for the scenes at 45° angles were higher than 90° scenes, with means of correlation values $\mu_R = 0.64$ and $\mu_R = 0.32$ for 45° and 90° scenes respectively. This is due to the large change in image content across the 90° camera baseline, making the actual number of similar image patches very low. This is exemplified by the suspiciously low correlation value of -0.2 for the 90° baseline example of the “Fashion 2” sequence shown in Figure 3.4 (bottom, right). This indicates that the assumption that detector performances is correlated to the overlap between similar and dissimilar distributions holds well for sufficiently similar image pairs, becoming less valid as the similarity between images decreases.

The correlation values in these experiments indicate a good relationship between K and percentage of correct matches. However, to allow meaningful comparisons between a pair of detectors, it is necessary to know whether a difference in K values between detectors corresponds to an increase in percentage correct matches. This is tested by evaluating the probability $P(p_1 > p_2 | K_1 > K_2)$, where p and K are the percentages and distribution separations calculated between pairs of ground truth images for all detectors. The results for the 45° and 90° scenes are $P(\cdot) = 0.78$ and $P(\cdot) = 0.67$ for the “Character 1” scene, and $P(\cdot) = 0.8$ and $P(\cdot) = 0.45$ for the “Fashion 2” scene respectively. This indicates that given an increase in K values between detectors, there is a good chance of an increase in the percentage of correct matches. More quantitative plots are given in Figure 3.5, where the difference between two K values is related to the percentage change in correct matches.

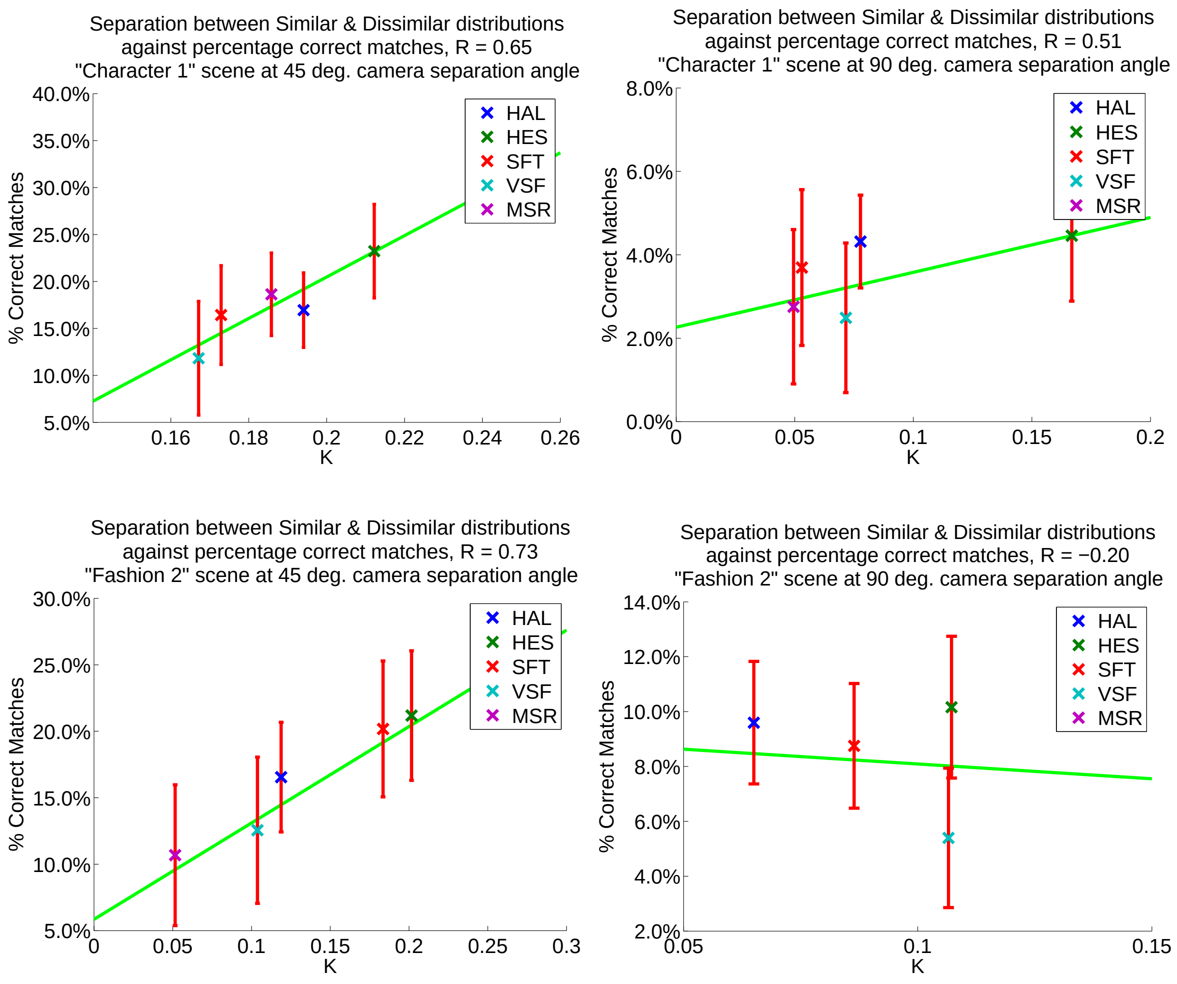


Figure 3.4: Experimental validation results for the ”Character 1” (top row) and “Fashion 2” (bottom row) scenes at 45° (left) and 90° (right) camera separation angles. The plots show the values of K calculated on distributions from various detectors against the percentage of correct matches found from ground truth. The height of the error-bars (red) from the mean are set to 1σ. The correlation coefficients R for the 45° and 90° camera angles are 0.65 and 0.51 for the “Character 1” scene, and 0.73 and -0.2 for the “Fashion 2” scene respectively. (See Section 3.3 for detector abbreviations.)

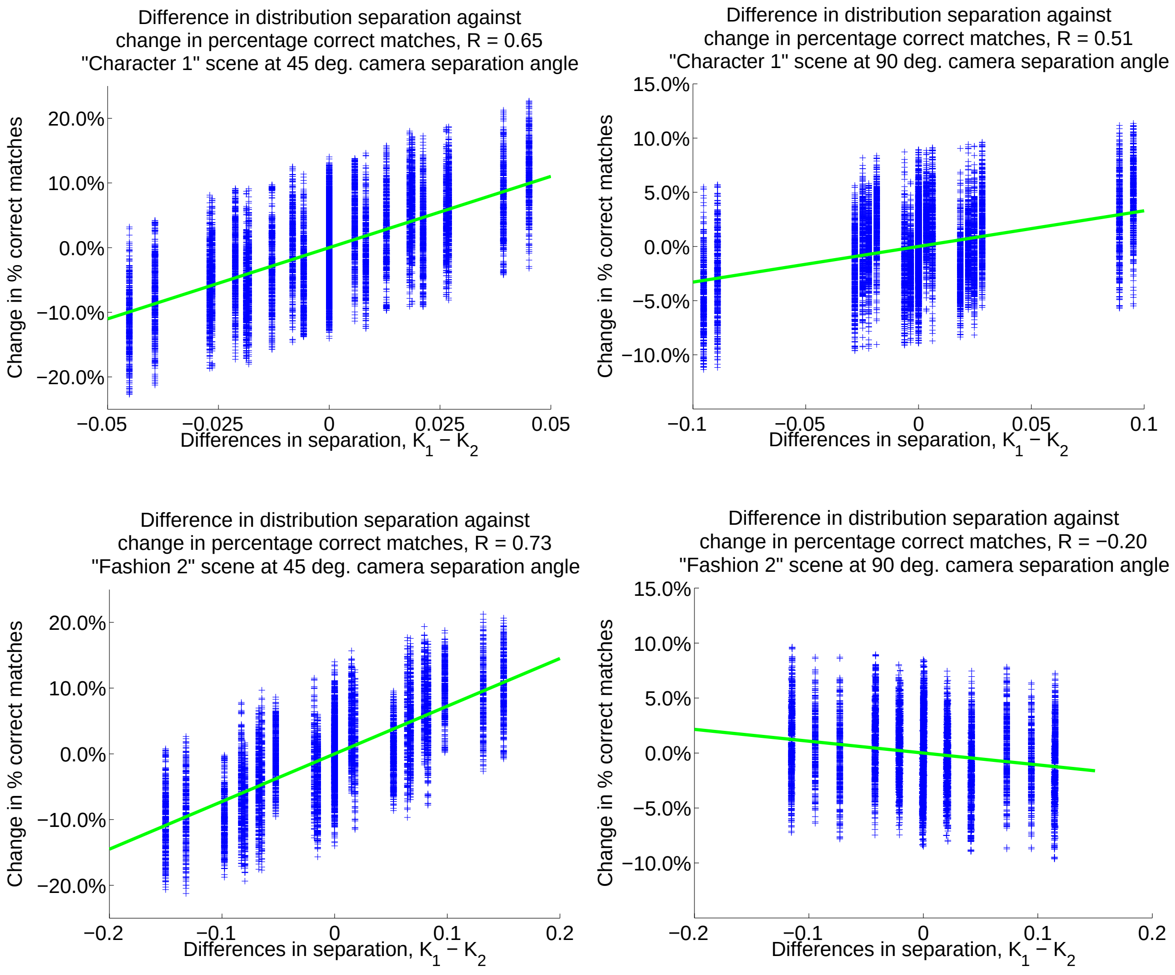


Figure 3.5: Change in percentage correct matches against the difference in measured K values for the 45° (left) and 90° (right) angles of the "Character 1" (top row) and "Fashion 2" (bottom row) scenes. The horizontal position of each blue "stripe" corresponds to the differences in distribution separation K values between all detectors, i.e. $(K_1 - K_2)\ \forall\ (\phi_1, \phi_2) \in \Phi$ for this scene. The vertical positions of the stripe samples are the changes in percentages of correct matches between all image pairs for all detectors. For example, one sample in the plot (i.e. a single blue cross), is comparing the difference in the percentage of correct matches for one image pair (vertical co-ordinate), against the difference in distribution separations for a pair of detectors (i.e. SIFT against MSER). The reason for the "stripe" grouping is that single distribution separation, K, values are calculated from the whole set of image pairs. These plots give an idea of the expected detector increase in detector performance given a difference in K values between two detectors.

3.3 Examples & Results

The following table shows the various detector combinations ϕ that comprise Φ for the example applications in this section. Only those parameter values most relevant to the performance of the detectors are given. An overview of many of these detectors can be found in Chapter 2. Additional details and default parameters used are available in the literature indicated. The

Detector	Parameters Used
Dual-Tree Wavelet (DTW) [50]	$\alpha = 0.1$, $\beta = 1/6$
Harris-Laplace (HAL) [114]	Corner threshold = 0.001
Hessian-Laplace (HEL) [114]	Hessian threshold = 0.01
SIFT (SFT) [105]	DoG space threshold = 0.02
Random Sampling (RND) [127]	Density = 0.05 feaures per pixel
Opponent Colour Harris-Laplace (OPH) [166]	Corner threshold = 0.85
VL-SIFT (VSF) [175]	DoG threshold 0.005
VL-MSER (MSR) [175]	$\delta = 1$

use of a “Random-Sampling” feature detector for object classification is shown by Nowak et al. [127]. In this work, it is used as a “control” detector to illustrate the effects of detecting point locations at random, and provide a base-line comparison for the other detectors. The detector parameters were set to give approximately the same number of features per image, this corresponds approximately to a density of 0.05 features per pixel. For consistency, the SIFT descriptor implementation of Vedaldi and Fulkerson [175] was used to describe the detected locations of all detectors in these examples. SIFT descriptors were calculated on the MSER regions according to the work of Forssén [56].

Object Recognition

In this example application, the user wants to be able to present an image of an unknown object to the system and classify it based on exemplar images of objects in a database. To simulate this object recognition scenario, images from the “Amsterdam Library of Images” (ALOI) [64] were processed by the system. Examples of the database images are shown in Figure 3.6. The ALOI “viewpoint” database contains 1000 object classes, each photographed in 360°, at 5° intervals. For this task, 40 objects were selected, and compared for all 72 angles. Each set of similar image pairs, A_m, for class m contains $({}^{72}_{2}P)$ pairs, resulting in $40 \cdot ({}^{72}_{2}P) = 204,480$ comparisons in the similar image pair set $A = \{A_1, \ldots, A_{40}\}$. The set of dissimilar image pairs $B = \{B_1, \ldots, B_{40}\}$ is filled by taking $40 \times 5,110$ pairs of random face images, ensuring that each pair of images does not belong to the same object or viewing angle.

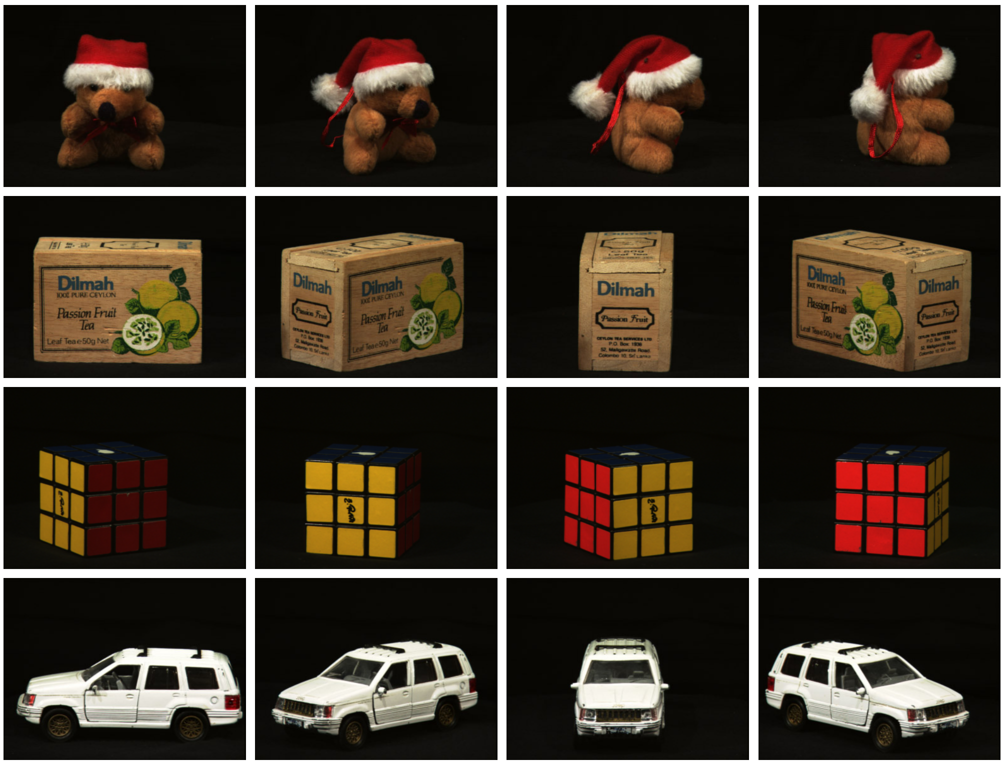


Figure 3.6: Examples of images from the Amsterdam Library of Images (ALOI) [64] at camera angles 45° apart, used in the Object Recognition application.

Face Detection

Faces from the IMM Faces database [165] were used for this example application. The proposed application is to be able to classify a face regardless of pose, facial expression or illumination conditions. Examples of faces from the database are shown in Figure 3.7. 40 people are photographed at 6 different head pose angles behind a blank background. For a given person, each pose image is compared to every other pose image for the same person, resulting in $40 \cdot ({}^{6}_{2}P) = 1,200$ image pair comparisons in the similar image pairs set $A = \{A_1, \ldots, A_{40}\}$. The set of dissimilar image pairs $B = \{B_1, \ldots, B_{40}\}$ is filled by taking 40×30 pairs of random face images, ensuring that each pair of images does not belong to the same face or angle.

Video Tracking

This example application proposes using features to track objects throughout a video sequence. Several shots from typical consumer video were analysed, examples shown in Figure 3.8. Given that the image content changes the least between two neighbouring frames, it is expected the matched distance distributions to be relatively low for all detectors between consecutive frames, yet high for the non-consecutive, dissimilar image content matches. For each shot, the fea-

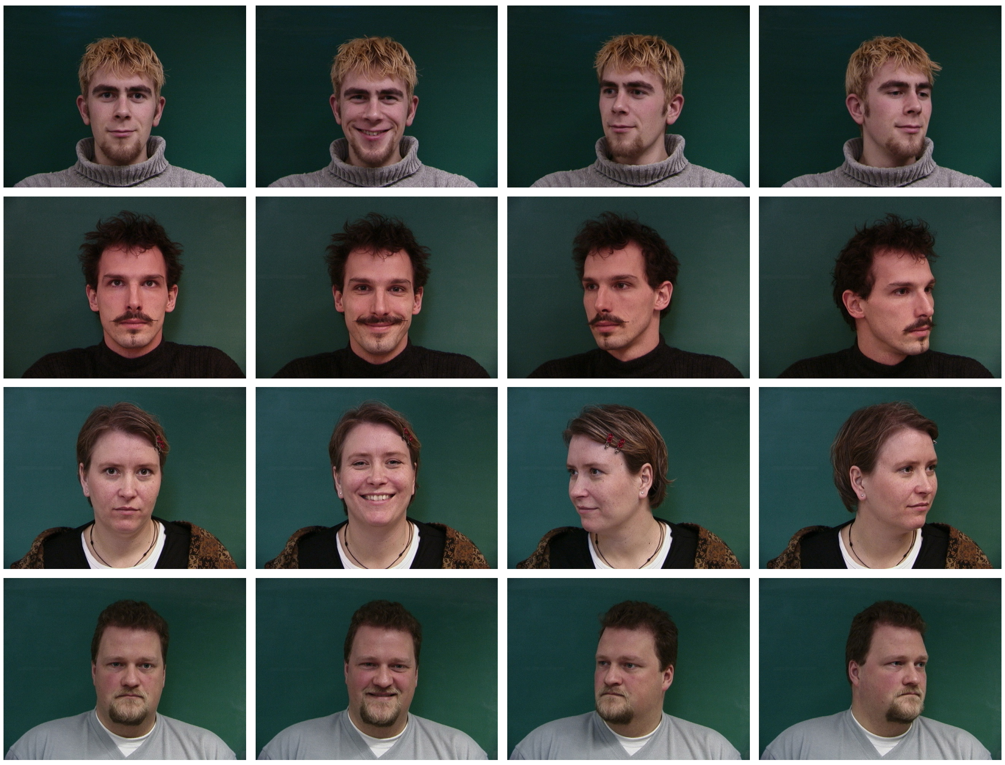


Figure 3.7: Example images of faces from the IMM Faces database [165], used for the Face Detection application. Each group of images contains the face of the same person at different angles and facial expressions.

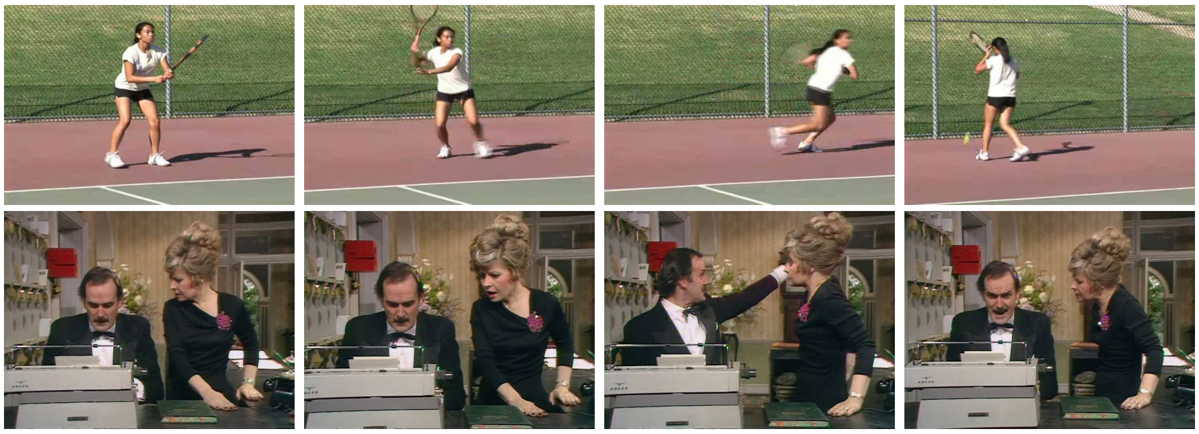


Figure 3.8: Example frames from Tennis and Fawlty-Towers video sequences used in the Video Tracking application. Detectors find correspondences between successive frames. Although the image content does not generally vary between frames, video presents different problems such as compression artefacts, interlacing, and most importantly, motion blur.

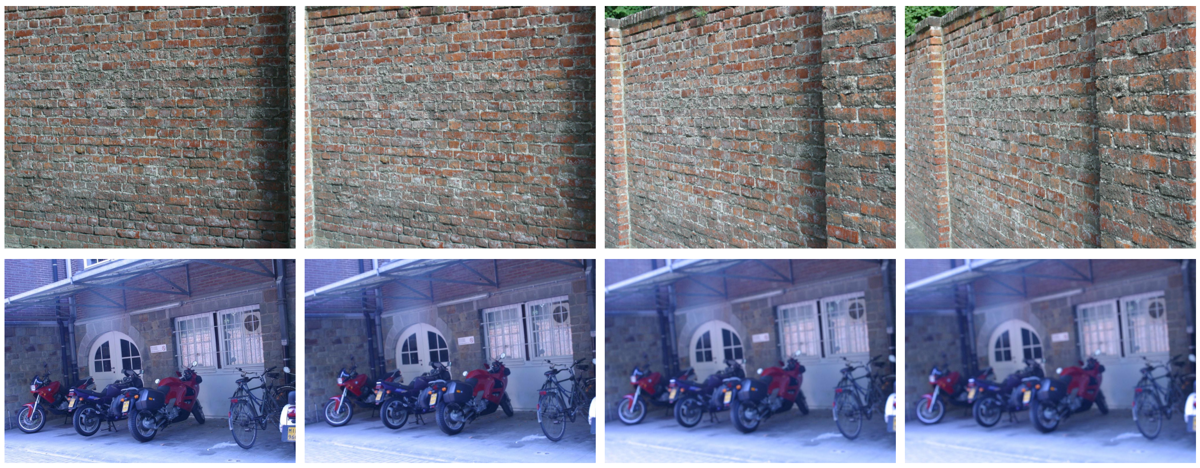


Figure 3.9: Example images from the Mikolajczyk and Schmid [117] scene database, and used in the Image Registration task. The content of all images in a scene is known to be highly similar. The challenge is to reliably re-detect that content throughout the scene in the presence of varying view points and photometric conditions, such as perspective transformations (top-row) and varying amounts of lens blur (bottom-row).

ture vector distance distribution is calculated from matching each frame of video to both the previous and next frame in the sequence. For this task, 5 shots of approximately 125 frames each were used, resulting in around $5 \cdot (125 \cdot 2) = 1,250$ image pairs in the similar image set $A = \{A_1, \ldots, A_5\}$. The dissimilar image set $B = \{B_1, \ldots, B_5\}$ is filled with 5×250 pairs of random, non-consecutive frames from other semantically different shots in the video.

Image Registration

The proposed task for this application is to register a pair of images. This involves estimating correspondences between image pairs at varying scales, rotations and image conditions, and subsequently warping the images to minimise the differences between the two. The images used for this application are the reference images used in the detector evaluation of Mikolajczyk and Schmid [117], examples are shown in Figure 3.9. The set is comprised of images of several outdoor scenes with various photo-metric conditions and from different viewpoints. Each image in a scene is compared to every other image in the same scene. For this task, 6 scenes with 6 images per scene were analysed, giving $6 \cdot ({}^6_2P) = 180$ image pairs in the similar image set $A = \{A_1, \ldots, A_6\}$. The dissimilar image set $B = \{B_1, \ldots, B_6\}$ is filled with 6×30 pairs of images from different scenes.

3.3.1 Example Application Results

For each class m in each application, features are calculated between each image pair in the sets, A_m and B_m. Correspondences are found between the image pairs, and the matched descriptor distances are used to estimate the distributions of descriptor distances for A_m and B_m. The separation between the distributions is then calculated using the Kolmogorov-Smirnov distance to give K_m. The results of the experiments of the example object, face and video tracking applications are shown in Figures 3.10 and 3.11, where $\mu_K = \frac{1}{M}\sum_{m=1}^{M} K_m$, and $\sigma_K^2 = \mathrm{Var}(K_m)$.

The results shown in Figure 3.10 give an overall idea of detector performance. Interestingly, the average K_m values are significantly different for each application. This probably corresponds to the degree of similarity between the compared images, for example, two consecutive frames of video (red) are probably more similar than faces at varying poses (green), giving overall higher separations. Also, the variances between classes are relatively low, indicating a strong agreement of K_m values between classes.

Notice that the relative separations between detectors are similar across the different applications. Of the non-random detectors, the HEL, HAL and MSR detectors perform well over all three applications in general, with the OPH detector performing poorly. Note that the objective of these examples is not to generalise the performance of the presented feature detectors for arbitrary applications, but to provide example results of the proposed protocol. Lastly, for all applications the distribution separations of the random control detector are very low, ≤ 0.1. This indicates that there is little difference between distributions of descriptors at randomly sampled feature point locations between pairs of similar images, and pairs of images with dissimilar content.

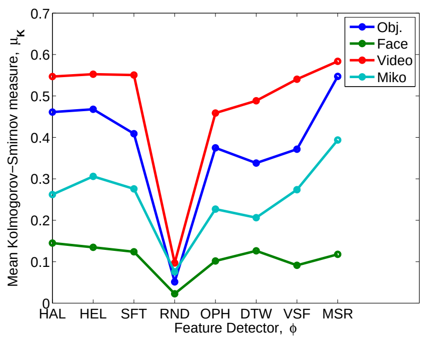


Figure 3.10: Mean distribution separations, μ_K, of example application datasets for various detector configurations, ϕ.

Det.	Obj.		Face		Video		Miko.	
	μ_K	σ_K^2	μ_K	σ_K^2	μ_K	σ_K^2	μ_K	σ_K^2
HAL	0.46	0.01	0.14	0.02	0.55	0.01	0.26	0.01
HEL	0.47	0.01	0.13	0.01	0.55	0.02	0.31	0.02
SFT	0.41	0.01	0.12	0.01	0.55	0.02	0.28	0.02
RND	0.05	0.00	0.02	0.00	0.10	0.00	0.08	0.00
OPH	0.37	0.01	0.10	0.01	0.46	0.02	0.23	0.00
DTW	0.34	0.01	0.13	0.01	0.49	0.02	0.21	0.01
VSF	0.37	0.01	0.09	0.01	0.54	0.02	0.27	0.01
MSR	0.55	0.04	0.12	0.00	0.58	0.01	0.39	0.01

Figure 3.11: Example application results, the means and variances of the distribution separations, μ_K and σ_K^2, are calculated over the K_m values for each object class, subject face, and video sequence.

3.4 Discussion & Conclusion

In this chapter a protocol was presented for measuring the performance of feature detectors on application specific image databases. The novel idea is to relate how well a given feature detection system performs to the separation between matched descriptor distance distributions from similar (intra-class) and dissimilar (intra-class) images. The proposed protocol is applied as a "black-box", making no assumptions on the content or structure of images, i.e. the target application, or camera configurations, and takes only the user's application image database as input. Results of the protocol calculated on typical feature-based applications were presented, and the protocol has been verified experimentally using ground truth from a realistic application.

Ideally, the performance of the whole application should be used as the criterion for feature detector assessment. However, many applications take a long time to process or train, making it infeasible to comprehensively evaluate many detector / parameter combinations by trial and error. This work proposes a systematic approach to evaluation that can look at the feature-based part of the application independently and say: "Detector X gives performance Y for image database Z" with some level of confidence. As the presented performance measure is based on one of the fundamental objects of most feature-based applications, the feature descriptor, it is expected that this measure is related to overall application performance.

However, the ground truth experiments illustrate the limits of using descriptor distance distributions to measure performance. That is, as the amount of similarity between images decreases, the correlation between distribution separation and performance also decreases, as shown by Figures 3.4 & 3.5. This trend indicates that the assumption of a relationship between K value and detector performance holds well for images exhibiting a reasonable level of similarity, and becomes less valid as the similarity between images decreases. For this reason, this protocol is not well suited to difficult image databases such as the Caltech 256 [67] or PASCAL [48] datasets, as the variability of the images within the object classes can be large. For example, two images in the "cup" class may contain a coffee mug and a child's beaker. Semantically these are both instances of cups, but the number of features with similar appearance will be quite low. In addition, these databases usually include a significant amount of "clutter" in the backgrounds of the images, again reducing the amount of similar local appearance between images in the same class. A measure of protocol suitability for given databases will be investigated in future work.

Chapter 4

Motion Cues for On-Line Event Parsing[1]

One-on-one coaching sessions are vital to sports training, where technique is improved and perfected through repetition. Sports such as cricket, tennis and golf have characteristic motion patterns associated with them, e.g. golf swings, tennis serves and cricket bats. During coaching sessions, these characteristic motions are repeated, and improved upon with the help of the coach. As consumer video equipment has become cheaper and easier to use, personal video recording has become a standard tool in coaching. However, the majority of training footage is manually edited or annotated for these interesting actions by coaching assistants. For example, the successful commercial software Dartfish [43] allows users to manually annotate and stroke the recorded footage to enhance the coaching session. The International Cricket Council (ICC) provides similar proprietary software that allows more detailed annotation of events specific to cricket, such as ball trajectories, landings and player statistics, but requires the intensive concentration of a skilled operator for good results.

This is a time consuming and tedious process, and in many scenarios, is usually performed after the match or training session. Automatic record and review of actions in sports training sessions is of great benefit to both coach and athlete. By analysing the motion of the player, it is possible to automatically detect and parse the coaching video for the characteristic sporting actions. This chapter presents a novel system to analyse implicit motion features to identify these characteristic motions and parse live coaching video, allowing the player and coach to record and review actions instantly, as shown in Figure 4.1. The proposed method avoids the intensive explicit computation of player silhouette and motion vector fields, allowing for a real-time, online application on standard hardware, significantly reducing the amount of manual effort required to annotate interesting sporting actions. Additionally, the accuracy of the motion

[1]Results from this chapter have been published as "Online Parsing of Sports Coaching Video through Intrinsic Motion Analysis" by Dan Ring and Anil Kokaram in *IEEE International Conference on Image Processing (ICIP)*, San Antonio, Texas, USA, September 2007.

parsing system is improved further by using local image features to compare the image content of detected actions, substantially reducing the overall number of false alarm actions.

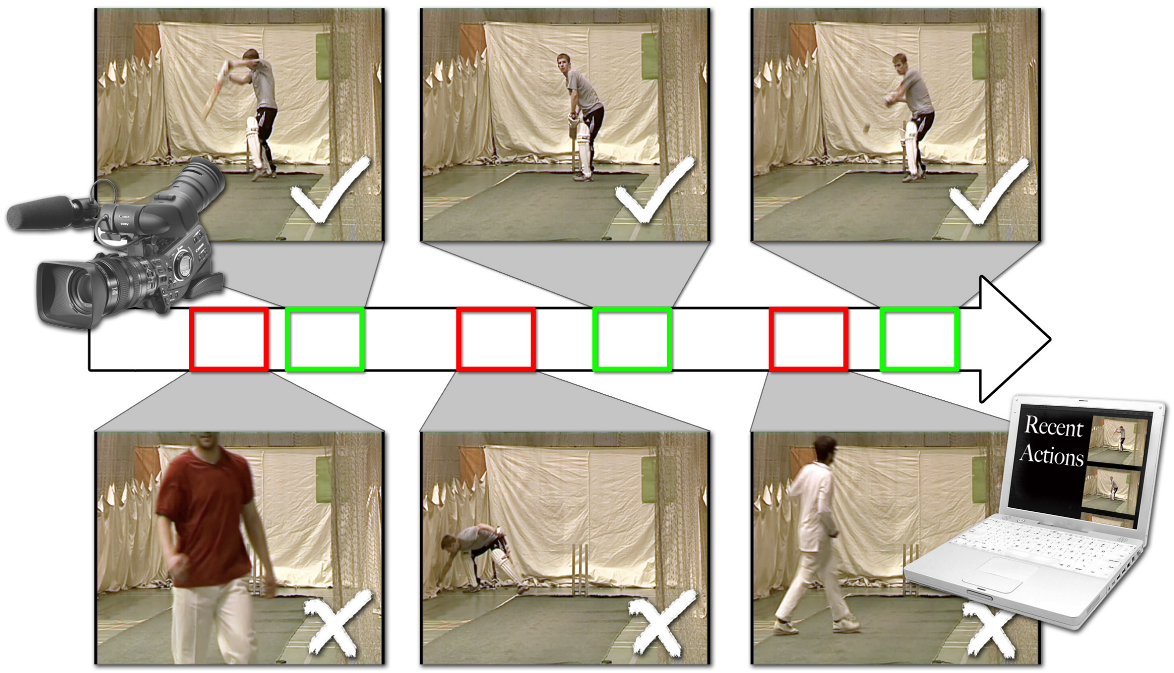


Figure 4.1: The objective of the application is to create a rapid record and review system. Video is parsed in real-time during the session for interesting actions. Examples of interesting actions are shown on top, with unwanted actions below.

The fields of sports analysis and motion-based video parsing are popular research areas. In Section 4.1, a selection of relevant work is presented to provide a context for the proposed application. The idea of intrinsic motion features is introduced and developed in Section 4.2, with the methods of parsing interesting events presented in Sections 4.3 and 4.4. Parameter selection is discussed in Section 4.5, results and discussion of the proposed system are given in Section 4.6. The feature based technique for identification of false alarms following motion cue detection is presented in Section 4.7, and a discussion of the entire system given in Section 4.8.

4.1 Previous Work

Automatic indexing and retrieval of sports video is a popular field of research [85, 185]. To focus the context of this review, the work discussed is limited to aspects directly related to the proposed application; *a*) using video for sports coaching, *b*) features for motion analysis, and *c*) event classification.

Sports Coaching and Video Processing

Of the many techniques for detecting sports actions from motion, the majority of sports coaching research involving video is focused on offline enhancement and quantitative evaluation. For example, Karliga & Ibrahim [80] extract the 3D motion of golf swings from single camera shots. A manual video segmentation step first performs background / foreground segmentation to isolate the human body. A simple 3D skeleton representing the proportions of the player is iteratively fit to the 2D segmentation, subject to geometric anatomic constraints such as joint freedom and location, and ensuring the model fit is temporally consistent over the action. The end result is 3D golf swing data that can be compared numerically against other players. Li et al. [96] propose a system for analysing the performance of diving athletes in two ways, “visually” and “biometrically”. For the visual analysis, global motion based mosaicing techniques are used to produce an informative story-board “panaroma” of the athlete’s dive into the water. In the biometric analysis, a human bounding box model is fitted to the athlete allowing diving postures to be measured quantitatively.

The 3D human data capture of Karliga & Ibrahim and the panorama visualisation of Li et al. are impressive post production review tools. However, a significant amount of manual interaction is required to produce results. In both systems, the video is first parsed manually before any post-processing takes place. The work in this chapter can be used to complement the performance analysis tools of Karliga & Ibrahim and Li et al. Using the proposed system to automatically parse the relevant sports actions, the amount of human interaction required can be minimised. The coach and athlete can be presented with recent actions, and can apply the post-processing on selected actions only, making the most of the time spent during the actual coaching sessions.

Features for Motion Analysis

Motion is a good feature for detecting events video, and has been used to great effect. Rea et al. [138] analyse the motion of the snooker balls to identify semantically interesting events, such as pots and fouls. The motion of the snooker balls is found by colour segmentation and particle filtering. Simple analyses of the motion paths are then used to accurately detect events such as collisions or ball pots. Although the use of colour segmentation and the criteria for detecting events are limited to snooker, Rea et al. present a simple and effective way to parse low level events from motion paths.

A more generic use of colour for event spotting is used by Ekin & Tekalp [47]. In their work on detection of generic sports events, Ekin & Tekalp note that shot boundary detection (video event parsing) is “usually the first step in generic video processing”. Ekin & Tekalp narrow the generic shot cut technique of comparing changes in colour histograms over time towards sports footage where a playing field is typically in view. Using prior knowledge of the colour of the playing field, the difference in occurrence frequencies of field coloured pixels between frames are used in conjunction with typical colour histogram differences to identify shot boundaries. The shot cut thresholds are adaptively set relative to the number of "field coloured" pixels in view, improving the accuracy and reliability of the cut detection. In the context of the application proposed in this chapter, the system of Ekin & Tekalp is probably better suited to finding the location of the player in the frame. For example, if the colour features of the athlete and background were analysed over time, the change in the number of "player pixels" might not be reliable enough for parsing, but the location of player pixels would be very useful for segmenting the player in the frame. Clearly colour is a useful feature for event detection, however it usually requires some user interaction to help model the player and background colour distributions.

Another effective way to detect interesting events in a generic sense is to look at the amounts of motion in a locality. The Caviar project [143, 133, 142] demonstrates the detection of unwanted activity in surveillance camera footage. Local motion fields are analysed for characteristic patterns indicating questionable activity, such as fighting and running. This work allows important footage to be brought to the attention of security guards, enabling more productive use of surveillance feeds. People in the footage are automatically detected, tracked, and coarsely segmented. A bank of classifiers are then trained on motion statistics from the local regions surrounding the tracked pedestrians. In the Caviar project, the work of Pla et al. [133] provides a comprehensive study into the choice of motion statistics, while Ribeiro et al. [142] detail the effects of different classifiers and their arrangements. Once trained, the system reliably classifies interesting actions within the surveillance video.

The objectives of the presented application overlap with those of typical motion detection systems, that is, using some understanding of motion to parse video for interesting events. The majority of the research in motion detection revolves around detection and classification of events in surveillance video. The most relevant work to the area of sports video is that of Pan et al. [130], where concepts from motion detection are applied to a generic approach for detecting slow motion shots in sports footage. The pixel-wise mean squared difference between field lines of consecutive frames is used as the motion detection feature. During slow motion, field lines are often repeated or dropped, causing large fluctuations in the difference feature, which can be captured by the numbers of zero crossings over time. A Hidden Markov Model (HMM) system is then trained on the zero crossing numbers, along with other measures of the difference feature, to reliably distinguish between slow motion events, commercials, editing effects and normal play. An interesting thing to note in this work is that the difference features are normalised over a temporal window, making the detection system more flexible.

Roh et al. [145, 146] propose a system for sports event detection in tennis using a "posture descriptor". Sets of representative postures are grouped in a particular learned order to compose a gesture, such as "right arm up, then down". By matching these posture sequences, higher level events such as hits and serves are detected. Shechtman et al. [154] introduce a local descriptor based approach to detecting sports events, where example images of interesting actions are presented to the system, and low resolution descriptors are calculated for each image based on the idea of self-similarity. For example, take a small image region, see how similar it is to all other regions of the same size in the image, and encode the locations of those similar regions relative to the current small image region. An image can then be represented by the concatenation of the most representative of the these descriptors. For sports, the idea is that a pose can be encoded (and later detected) by the particular arrangement of similar image patches, i.e. a raised arm is expected to have many similar image patches arranged vertically. By extending this self-similarity to a 3D space-time volume of video, interesting pose transitions (or events) can be detected against a supplied example event.

The application presented in this chapter requires rapid record and review on "off the shelf" hardware. The works of Roh et al. and Shechtman et al. are relevant to the proposed sports action detection system because they are both computationally efficient and can be extended to other sports by changing the training data. For the proposed application however, the user should not need to train the system in advance or manually specify any athlete or background colour distributions. This rules out using the event spotting systems of Rea et al. and Ekin & Tekalp. The results of the Caviar system are impressive, but the statistics rely on the calculation of a large motion vector field. Real-time calculation of motion vectors for full HD resolution video is possible on some hardware, such as recent high-definition TVs, however there is still value in exploring methods of motion parsing with low computation requirements. For example, for use in hardware such as cameras or mobile phones. The appeal of the work in Caviar is that various statistics, based on the motion field, are used to accurately classify actions. It is clear that some of the statistics can be approximated without needing the full motion vector field, e.g. an *intrinsic* motion feature, such as the amount of motion surrounding the pedestrian. This is exemplified by the work of Lie et al. [97], where dense motion vectors are calculated on consecutive frames, only to have the magnitudes of the vectors averaged to produce the 1D signal on which the interesting baseball events are detected. Although the detection results of Lie et al. are reasonably good, it is clear that an *intrinsic* motion feature could be substituted in place of the dense motion vectors. The work in this chapter develops an intrinsic motion feature, and shows that it can be used for successful parsing of sports coaching video.

Classifying Interesting Events

The work of Li & Sezan [94] first define two paradigms for automatically detecting interesting events, "deterministic" and "probabilistic", before discussing the merits and weaknesses of typical approaches of each. Both paradigms use the same image features to model the likelihood of an event happening, such as particular colours, dominant motion, or temporal discontinuities. Li & Sezan posit that deterministic approaches are often easy to implement and computationally efficient, yet are usually predicated on a series of hard-coded conditions, with equally rigid, hand-picked thresholds. Once configured for one sport type, it is difficult to adapt the system to another sport. Instead, the probabilistic approach assumes a (known or unknown) number of "states" based on actual physical situations, such as "serve", "rally" or "foul" in tennis. Probabilities are then assigned to each of the states based on observations in the feature spaces. Prior information about state transitions is then incorporated, typically using some sort of Markov Random Field constraint in a state machine model (e.g. Hidden Markov Models (HMM)), and solved to find the most likely sequence of states throughout the video. Li & Sezan believe that probabilistic approaches are to be preferred, offering greater flexibility, and show that for their own real-time event detection application, their probabilistic algorithm provides higher accuracy. In their survey work, Kokaram et al. [85] also posit that most of the interesting work in sports summarisation use HMMs for event classification.

Continuing with the idea of flexibility, the work by Lu & Tan [106] proposes a system for automatically classifying shots within sports footage using unsupervised clustering of shot colour histograms (SCH). Here the word "shot" is used to refer to contiguous segments of video, as opposed to a sports action. The interesting aspect of this work is that the appearance and number of unique "shots" is not known a priori. Their shot cluster centroids are found by maximising the inter-class SCH distances and minimising the intra-class SCH distances according to Fisher's linear discriminant analysis [54]. The results of Lu & Tan indicate that the number of shots in sports video can be clearly identified. Once the shots are identified by their SCH centroid, it is then possible to identify "interesting" actions within the shot, by measuring the individual frames departure from its assigned centroid. The work of Lu & Tan demonstrates that video can be successfully partitioned at various granularities, such as "shot" and "action-within-shot" automatically. Work by Gao & Tang [61] uses clustering techniques to perform similar shot detection and classification on long ($>$ 5 hours) of news footage.

Although the works of Lu & Tan and Gao & Tang show that video can be successfully parsed unsupervised, they are intended for off-line processing. Many of the previously discussed works have elements that would be useful for the intended record & review application, for example the on-line motion learning of the Caviar project, or the unsupervised shot-detection of Lu & Tan, and the probabilistic parsing framework of Li & Sezan. However, no single strategy fits all of the requirements for the desired application. The following work incorporates and adapts some of these concepts resulting in a novel system for real-time parsing of sports coaching video that is able to adapt and operate "out-of-the-box" with any sport actions that exhibit a repetitive nature.

4.2 Analysing Intrinsic Motion

Many sports have particular actions associated with them, for example baseball pitches, or tennis serves. Such actions lend themselves well to on-line "record-and-review" sessions; as the athlete does not need to travel large distances during the action, the camera position can be fixed and does not need an operator. This unattended and controlled set-up is ideal for video processing as the only motion present should be that of the athlete and coach, and not of the camera. The

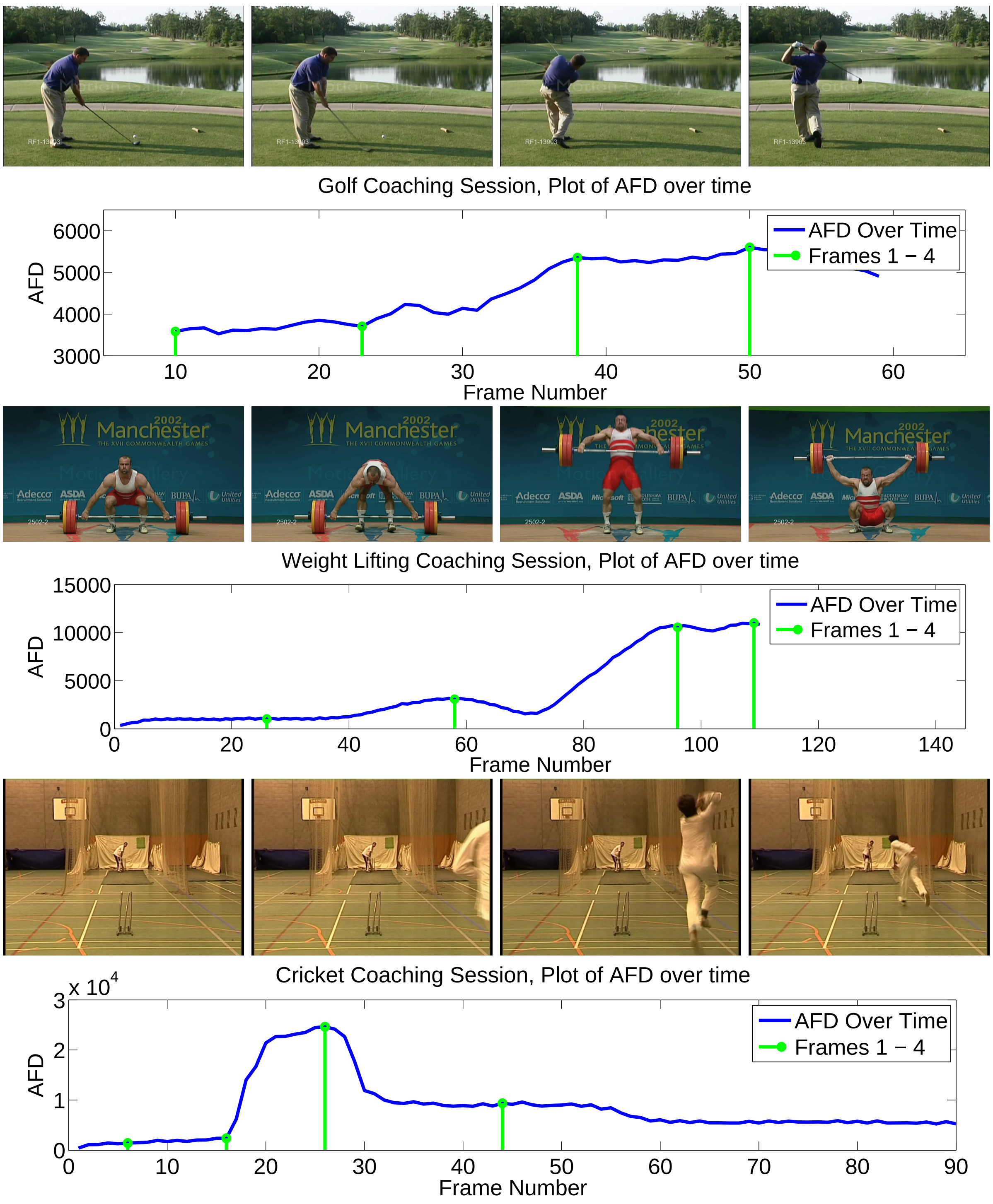


Figure 4.2: Typical "Ready-then-Action" motion patterns of a golf training session (top), weight lifting (middle) and cricket bowling (bottom). The motion pattern begins with the athlete getting ready, anticipating what is to follow, and is usually stationary or moving fairly little (far-left and centre-left images). When the action occurs, there is a sudden burst of motion (centre-right and far-right images), before returning to the "ready" state. The motion for these examples is calculated by the *absolute-frame-difference* (AFD), $\sum_{\mathbf{x}\in\mathcal{X}} |I_n(\mathbf{x}) - I_{n-1}(\mathbf{x})|$

task now is to develop a set of motion features that can be used to accurately parse the video for interesting actions.

The repetitive characteristic motions in coaching video often exhibit a “ready-then-action” pattern, that is, periods of minimal movement while setting up the shot etc., followed by a burst of motion. This can be seen in the coaching video examples shown in Figure 4.2. The objective is to select the motion features that can reliably and efficiently detect these “ready-then-action” patterns to signal the application to begin and end recording of the actions. Obvious motion features to use are dense motion vectors from full frame motion estimation, sometimes known as optical flow. However, these accurate motion vectors have a high computational burden associated with them [86]. Efficient implementations of motion estimators have long occupied the interest of the image processing community. Of particular relevance to the presented work is the use of integral projections [118], enabling real-time motion estimation and compensation applications [39, 144]. For a frame n, of size $[M \times N]$, the integral horizontal and vertical projections are

$$\rho_{h,n}(i) = \sum_{v \in \{1,\ldots,N\}} I_n(i,v)$$
$$\rho_{v,n}(i) = \sum_{h \in \{1,\ldots,M\}} I_n(h,i)$$

where $I_n(x,y)$ is the intensity of the n-th image at location (x,y). In the context of estimating dominant motion in the frame, projecting the 2D image data onto the horizontal and vertical axes reduces the computational cost from $O(MN)$ to $O(M+N)$ for an image. For example, when calculating the difference between the images in each iteration of the gradient-descent approach of motion estimation, only 2 vectors of size M and N respectively need to be compared, instead of the MN size image. For any reasonable sized image (PAL resolution or higher), the computational saving is significant. Examples of image projections are shown in Figure 4.3. Considering the fixed camera set-up and single player in view constraints of the proposed application, projections appear useful for calculating motion features.

4.2.1 Analysing Movement

In the work of Crawford et al. [39], the apparent horizontal and vertical motion is calculated by a multi-resolution gradient-descent approach, resulting in a 2 element vector defining the estimated translational shift between a pair of consecutive frames. Although the method proposed by Crawford et al. [39] is good for estimating the dominant motion for moving camera scenes, where most of the frame undergoes the same motion, it does not capture the small, local motions exhibited by the athlete very well[2]. This is usually because the influence of the large stationary background encourages the motion toward zero, or someone has entered the near foreground and

[2]Robinson & Milanfar [144] state that estimating motion on a set of 1D projections is more suited for global motion.

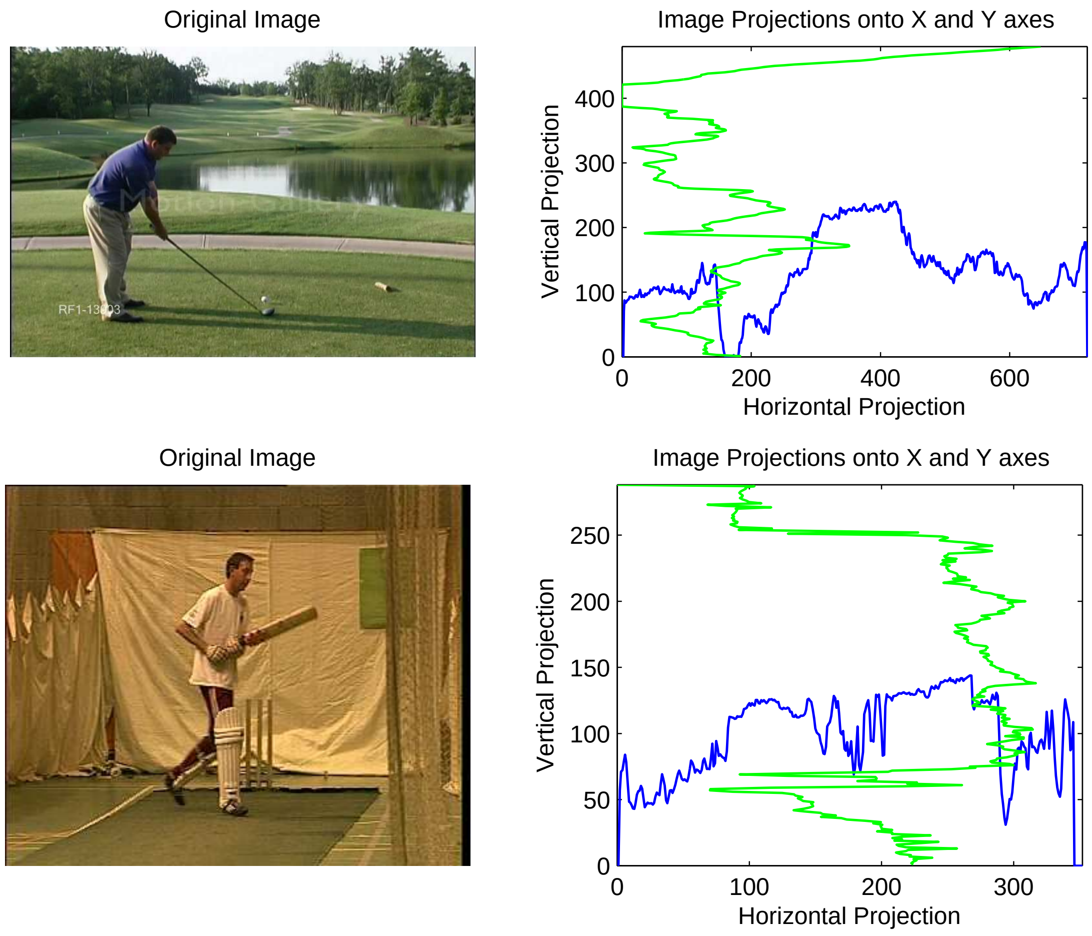


Figure 4.3: Example of image projections for golf (top) and cricket (bottom) coaching scenes. The horizontal and vertical projections (scaled for illustration) are shown in blue and green respectively. High values in the projections correspond to to bright regions. For example the square background net in the cricket scene becomes a plateau in both the horizontal and vertical projections.

by occupying a larger proportion of the image frame over-shadows the motion of the athlete. Another problem is that complicated local motions are averaged out. For example, during the swing of a cricket bat, the player might bend their knees downwards, while swinging the bat upwards, causing the net calculated motion to be close to zero. The latter contrived example could be accounted for by estimating changes in scale as well as translation, however the point is that typically complex sports motions are difficult to capture using the 2D to 1D estimation scheme of Crawford et al.

Consider instead, the proposal that simply looking at the *amount of movement* is sufficient for the purposes of accurately parsing video for actions automatically. The idea is that the magnitude of apparent movement is more useful for identifying actions than the translational displacement vector. The inter-frame difference is good measure of apparent movement, that is, the absolute difference between two horizontal projections of consecutive frames, $|\rho_{h,n} - \rho_{h,n-1}|$, gives a reasonable idea of the amount and horizontal location of movement in the scene. This proposed measure is dubbed an *intrinsic* motion feature, as the interesting information from player movement is found without the need to explicitly calculate motion vectors. To clarify what the distinction between “movement” and “motion” is in this context; motion is the recovery of (apparent) correspondences between a pair of consecutive frames, while movement is the real-world process that causes motion. Movement is therefore *related* to motion, but does not necessarily require motion vectors to be calculated.

The inter-frame difference is a powerful measure of movement, as evident by its use in the gradient-descent based motion estimation equation, the 1D example of which is used by Crawford et al. However, the inter-frame difference on its own does not distinguish between athlete specific movement and other movement in the scene. For example, if there are multiple subjects exhibiting motion in the shot, as is often the case when the coach is discussing technique or a team-mate walks in front of the camera, the projection difference of Equation 4.1 naturally includes that of the “uninteresting” subject. For this reason, it is first necessary to isolate the area in which the athletic motion is likely to occur by identifying a “region of interest” on which to estimate the amount of movement.

Consider the absolute difference, $\Delta\rho_k$ between the projections of two consecutive frames about the horizontal and vertical axes

$$\begin{aligned}\Delta\rho_{h,n} &= |\rho_{h,n} - \rho_{h,n-1}| \\ \Delta\rho_{v,n} &= |\rho_{v,n} - \rho_{v,n-1}|.\end{aligned}$$

Given some movement between the pair of frames, distinct peaks in the difference projections $\Delta\rho_k$ will be observed. Consider the case where the player moves from left to right in a pair of frames. During this motion, the athlete will occlude background pixels on their right side, while uncovering pixels on their left. Assuming the pixel colour intensities of the athlete are distinguishable from those of background pixels, this will usually induce two or more peaks in each $\Delta\rho_k$ projection. An example of the peaks in the horizontal projection difference caused by

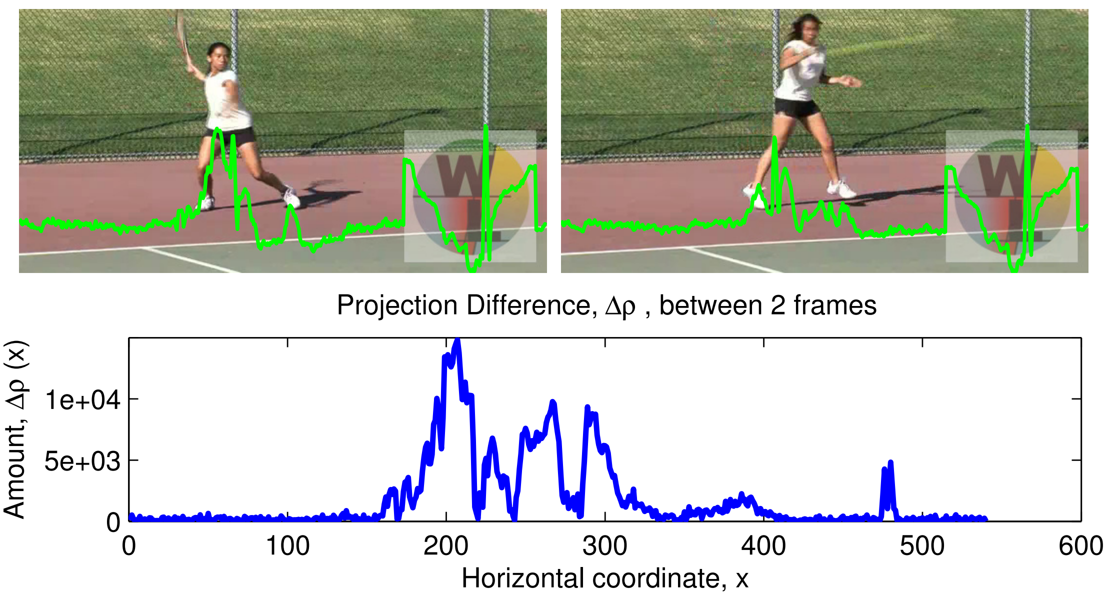


Figure 4.4: Example of peaks in the projection difference, $\Delta\rho(x)$ (blue, bottom row), due to player movement between projections of a pair of frames (green, top row). The two peaks on the right of the cluster (with modes around 260 and 400 resp.) were induced by the movement of the athlete's body and left leg, while the far left peak is the result of uncovering the pole in the background. The small peak to the far right of the plot is a spurious peak caused by very slight camera movement.

player movement is shown by the example in Figure 4.4.

Although the peaks of the $\Delta\rho_k$ projections themselves are not useful, the player motion can be localised well by looking at the centre of mass of $\Delta\rho_k$. If the projection difference $\Delta\rho_k$ is thought of as a probability distribution of likely motion locations X, the centre of mass corresponds to calculating the expected value of the projection indices $x \in X$, of $\Delta\rho_k(x)$. With the location of the player motion identified, the scale (width and height) of the player motion region needs to be found. Again, using the data of $\Delta\rho_k(x)$ as a distribution, the variance of the locations X gives a good estimate of region size. The player motion location (centre of mass), and scale (variance), are given by $E[\Delta\rho_{k,n}(X)]$ and $E[(\Delta\rho_{k,n}(X) - \mu_k)^2]$ respectively, where $E[]$ is the expectation operator. This will calculate the location and scale of motion for a single pair of frames. This is easily extended to model the region of interesting motion over time by averaging the difference projections over time, for example over N frames to give $\overline{\Delta\rho_k} = (\sum_n^N \Delta\rho_{k,n})/N$. The mean and variance of the temporally averaged difference projection, μ_k and σ_k^2, are used to give the location and scale of player motion over several frames,

$$\mu_k = E[\overline{\Delta\rho_k}(X)] = \frac{\sum_x x\overline{\Delta\rho_k}(x)}{\sum_x \overline{\Delta\rho_k}(x)}$$

$$\sigma_k^2 = E[(\overline{\Delta\rho_k}(X) - \mu_k)^2] = E[\overline{\Delta\rho_k}(X)^2] - \mu_k^2 = \frac{\sum_x \left(x\overline{\Delta\rho_k}(x)\right)^2}{\sum_x \left(\overline{\Delta\rho_k}(x)\right)^2} \ .$$

To get a good idea of where the action is taking place, the number of frames N used to calculate

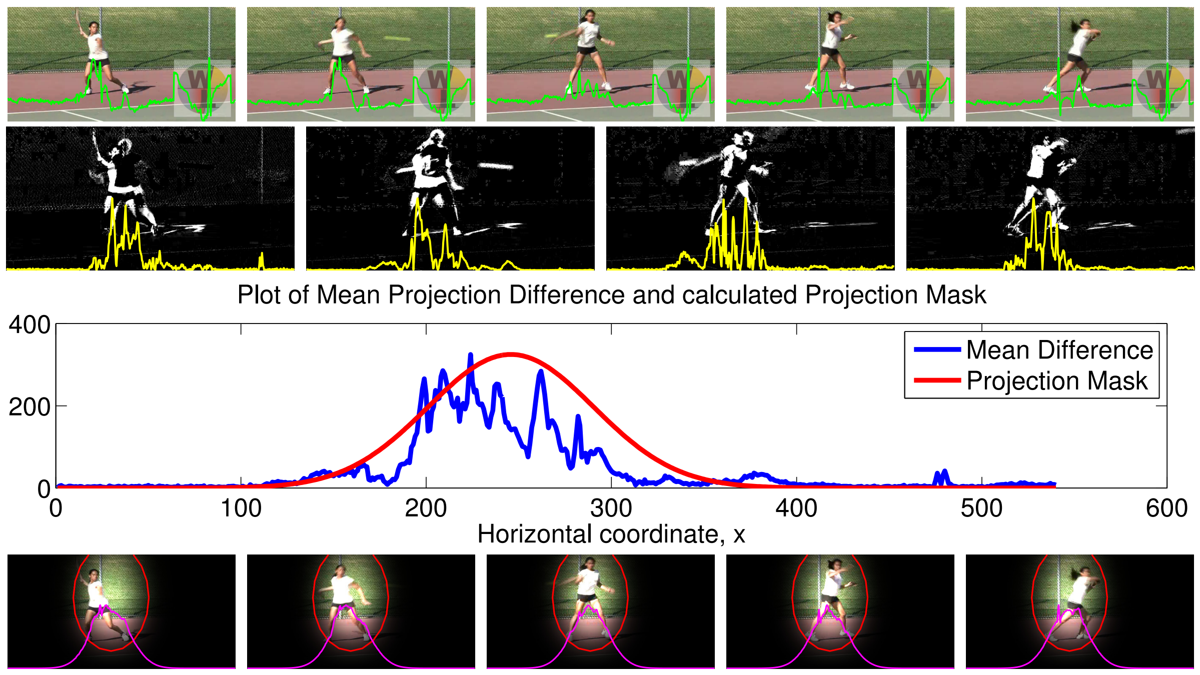


Figure 4.5: Example of projection mask calculation. Top row, $N = 5$ consecutive frames and their associated horizontal projections (green). Second row, the differences between consecutive projections are shown in yellow on top of the 2D absolute frame difference for illustration. A pdf of probable locations of player movement is given then by the mean of the differences (third row, blue). The projection mask (third row, red) is calculated using the mean and variance of the pdf as the location and scale of the mask respectively. The player region in the original projections are then masked (bottom row, magenta), with a 2D back projection of the horizontal and vertical masks applied to the original images for illustration. The width and height of the red ellipse is set to twice the horizontal and vertical variance, $[2\sigma_h, 2\sigma_v]$, of the distribution.

$\overline{\Delta\rho_k}$ should be high enough to capture a few actions. Sensible values for N are discussed later on in Section 4.5. The task now is to use the location and scale of probable player motion to weight the original projections, $\rho_{k,n}$. Given that the centre and the variance of the player location have been found, a simple Gaussian shaped weighting is applied to the original projection to give the weighted projection,

$$\hat{\rho}_{k,n}(x) = \rho_{k,n}(x)\exp\left(-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right) . \tag{4.1}$$

As only the relative change in values of the weighted projections between two frames, $\hat{\rho}_{k,n-1}$ and $\hat{\rho}_{k,n}$, are important, $\hat{\rho}_{k,n}$ does not need to be normalised, and the factor, $\frac{1}{\sqrt{2\pi\sigma_k^2}}$, usually associated with the Gaussian distribution function is omitted. A worked example of the projection weighting scheme, including the back projection of the 1D mask onto the original 2D frames is shown in Figure 4.5. Although they are not explicitly used in this work, the estimated parameters of the Gaussian distribution themselves can be useful when the temporal window N is sufficiently small. For example, μ_k provides a reliable player track of where the centre of the athlete is in the frame. Using knowledge of the scene, this could be used to only allow detections in a specific region of the image, i.e. only in front of the net, or between a pair of goal-posts. With the weighted projections $\hat{\rho}_{k,n}(x)$ focused on the area of the player, the sum of the difference between projections of consecutive frames is a reliable measure of how much the athlete is moving,

$$u_k(n) = \sum_{i=0}^{N_k} |\hat{\rho}_{k,n}(i) - \hat{\rho}_{k,n-1}(i)| \tag{4.2}$$

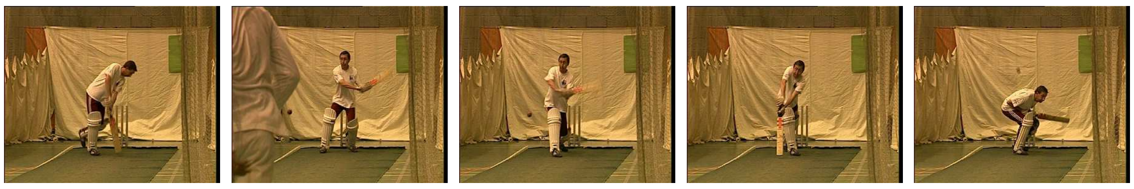
where n is the current frame number, N_k is the number of elements in the projection for dimension k (horizontal or vertical). With the methods for isolating and estimating apparent player movement in the scene defined, their use in identifying interesting actions is now discussed.

4.3 Analysing the Movement Signal

Recall from Figure 4.2 the distinctive movement patterns apparent in many sports coaching videos. The presented examples exhibit clear, dramatic increases in the amount of movement present which should be simple to segment. The question is, what amount of movement is necessary to signal an interesting event? In real world scenarios, there are many factors that influence the size and shape of the movement pattern. For example, the player's on-screen position; the closer the player is to the camera the more dramatic a given movement appears. The type of sport also affects the movement pattern, golf strokes often have two peaks of high movement corresponding to the back and forward swings of the club, whereas cricket bats will often only have one movement peak. The fitness and alacrity of the player are other factors that contribute to the shape of the movement pattern. These real-world details mean that predicting the shape of player movement amount in advance is a difficult feat in itself. Although it might be possible to train the system for a particular athlete performing a particular action, much like speech recognition software, an important requirement of the application is to keep user interaction to an absolute minimum. The only useful observations that are constant regardless of the shapes of the movement pattern are; 1. that there is some *relative* increase in movement before and during an interesting action, and 2. on average, the time spent performing an action is much less than the time between actions. This high-level knowledge that will be used to detect events.

Some examples of player movements will now be looked at in detail using the weighted projection based estimation scheme. The example in Figure 4.6 shows an 1800 frame clip of a cricket coaching session containing 5 interesting actions, the frame ranges of each of the 5 actions are denoted by the ground truth plot (red). However, there are still many peaks induced by apparent movement in the frame that are not relevant to the user; a high amount of movement does not necessarily mean an action has happened. A method for dealing with these high-movement, yet irrelevant actions is needed.

Observe that when an action occurs in Figure 4.6, the apparent movement increases dra-



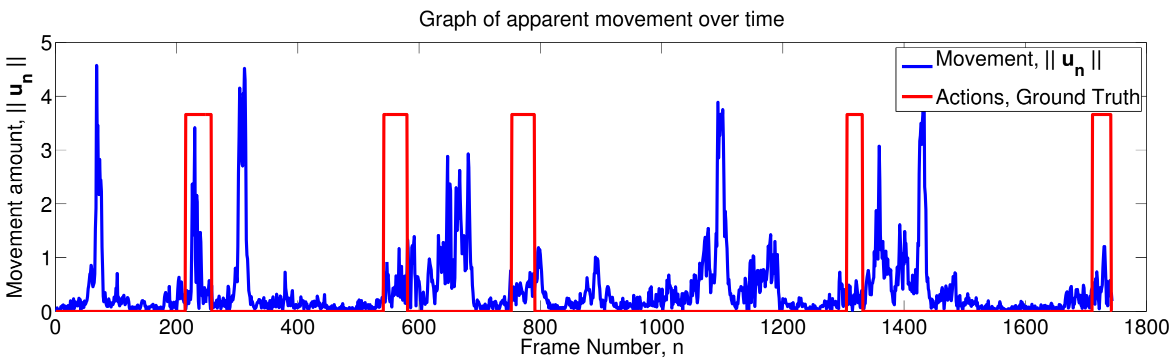

Graph of apparent movement over time
Movement, || $\mathbf{u}_n$ ||
Actions, Ground Truth
Movement amount, || $\mathbf{u}_n$ ||
Frame Number, n
0
1
2
3
4
5
0
200
400
600
800
1000
1200
1400
1600
1800

Figure 4.6: Example of athlete movement over time during a cricket batting session. Five actions (top row) have been identified over 1800 frames, shown in yellow plots in middle and bottom plots.

matically from some previously "low" value. It is not necessary to know the exact "high" and "low" values to detect an event, just that a sharp change from any "low" value to a "higher" value probably means something interesting has happened. How "low" or "high" the current movement value is can be estimated by collecting previous movement values over a temporal window. To give an idea of the range of expected movement, previous movement values are sampled on-line over a temporal window size of M frames,

$$\mathbf{v}_n = \left[\|\mathbf{u}_{n-M}\|, \|\mathbf{u}_{n-M+1}\|, \ldots, \|\mathbf{u}_{n-1}\|, \|\mathbf{u}_n\| \right]^T$$

where $\mathbf{v}_n$ is a column-vector of length M. In this case, the amount of movement between a pair of frames is given by the magnitude of the movement vector, $\|\mathbf{u}_n\|$, calculated by the magnitude of the horizontal and vertical components of Equation 4.2, where $\mathbf{u}_n = [u_{h,n}, u_{v,n}]$ for frame n. Instead of trying to model all possible magnitudes and shapes of athlete movements, it is assumed that recent movement values in the summary vector v_n can be used to say how likely the current frame is to belong to an interesting action based on its movement value. In this way, the system has no "memory" after M frames. The choice of value for M is therefore important, and will be discussed in detail later on. At the very least, M should be large enough to capture one action in order to determine what a "significant" increase in movement is.

The likelihood that an observed movement value $\|\mathbf{u}_n\|$, in the current frame n, is "high" can be estimated by using the vector of previous movement values $\mathbf{v}_n$. Given the vector $\mathbf{v}_n$ at frame n, the significance is calculated, i.e. probability of observing a movement value equal or less than $\|\mathbf{u}_n\|$,

$$P(X \leq \|\mathbf{u}_n\|) = \int_0^{\|\mathbf{u}_n\|} f(x)\, dx \tag{4.3}$$

where X is the random variable of all possible movement values in $\mathbf{v}_n$, f is the probability distribution function (pdf) of the movement values in $\mathbf{v}_n$, $f(x)$ is the probability of the value $x \in X$ occurring in $\mathbf{v}_n$. Note that this probability is calculated for the current frame n only, for the next frame the values inside $\mathbf{v}_{n+1}$ will be used instead.

For the sake of argument, assume that the movement values in $\mathbf{v}_n$ are Gaussian distributed, the simple Normal test statistic for the frame n is given by $\frac{\|\mathbf{u}_n\| - \overline{\mathbf{v}_n}}{\sqrt{\mathrm{Var}(\mathbf{v}_n)}}$. The probability that a movement observation of value $\|\mathbf{u}_n\|$ arose other than chance can then be calculated by finding the value of the cumulative Gaussian distribution at $\|\mathbf{u}_n\|$,

$$P(X \leq \|\mathbf{u}_n\|) = \frac{1}{\sqrt{2\pi \mathrm{Var}(\mathbf{v}_n)}} \int_0^{\|\mathbf{u}_n\|} \exp\Big(\frac{u - \overline{\mathbf{v}_n}}{\sqrt{\mathrm{Var}(\mathbf{v}_n)}}\Big) du \ .$$

In most cases the distributions of values in $\mathbf{v}_n$ are not well represented by the Normal distribution, as shown in Figure 4.7 (top, right), and can generally not be assumed to belong to any particular distribution. Instead of trying to model the values in $\mathbf{v}_n$, the significance of frame n is measured empirically using the cumulative distribution function (cdf.) of the values in $\mathbf{v}_n$.

To see the effect of calculating the likelihood on temporally windowed data, observe Figure 4.7. Shown in Figure 4.7 (top left) are two distributions of calculated movement amounts, one of all movement values throughout the sequence (green), the other of movement values from frames labelled manually as belonging to an "interesting" action (yellow). As can be seen, there is no way to clearly separate the first distribution into "interesting" and "not-interesting" with a single threshold. The "significant window" data, $\mathbf{v}_n$ attempts to improve this. Observe Figure 4.7 (top, right). From the start to the end of the first action (between frames 220 and 260 of Figure 4.6), the lobe of the distribution of values in $\mathbf{v}_n$ has shifted higher towards 1 (i.e. from the brown to tan distributions), revealing a noticeable difference between the distributions particularly around the tails (from 0.7 and higher). This indicates that movement values before an action are generally lower than movement values during an action, which is expected. This separation can be seen more clearly by the pair of cdf's for two vectors $\mathbf{v}$ shown in Figure 4.7 (bottom, left), each derived from the previous $M = 90$ frames. In Figure 4.7 (bottom, right), the distributions of the movement *likelihood* (i.e. $P(X \leq \|\mathbf{u}_n\|)$) for interesting (ground truth, yellow) and not-interesting (green) actions are better separated. The effect of this is that a global threshold is able to capture the interesting actions better.

Calculating the likelihood over all frames using the probability assignment of Equation 4.3 results in a normalised movement signal, as shown in Figure 4.8 (bottom). At first glance, there might seem to be little or no improvement between the before and after plots of Figure 4.8. However, looking at the plot of the movement likelihood over time, it can be seen that all the interesting actions have values of $P(X \leq \|\mathbf{u}_n\|) \geq 0.7$. With a single global threshold it is

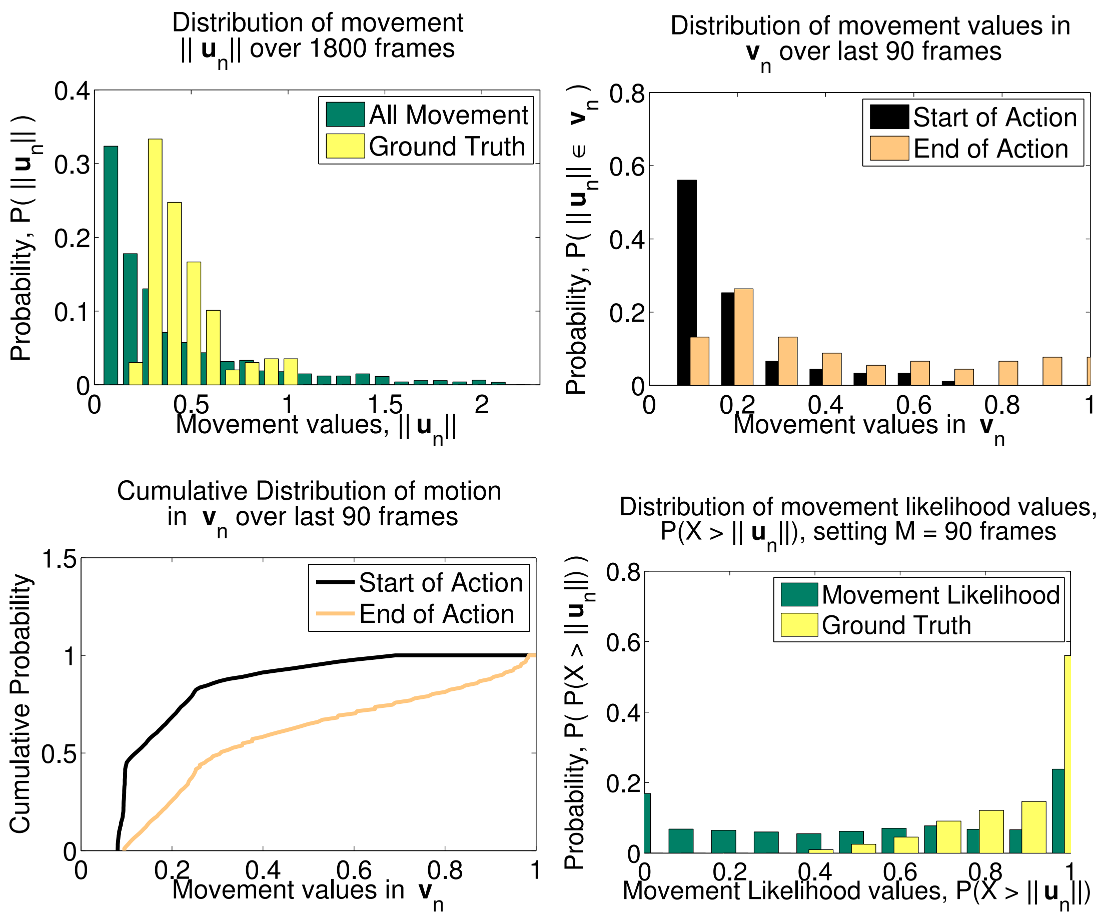


Figure 4.7: Movement value, and movement likelihood distributions from a portion of cricket batting session video.

possible to capture all the relevant actions[3]. The task now is to use the movement likelihood, $P(X \leq \|\mathbf{u}_n\|)$, to reliably detect the interesting actions.

4.4 Identifying Interesting Player Actions

Recall that the two high-level features proposed for detecting events are; a relative increase in the amount of movement, and that the time during and event is much less than the time between events. Analysing recent movement values in a window allows a likelihood of "interesting" movement, $P(X \leq \|\mathbf{u}_n\|)$, to be calculated. Relative changes in movement amounts can then be found using a global threshold on the likelihood. This movement likelihood encodes the first high-level criterion for an interesting event. However, thresholding the movement likelihood

[3]Note that this does not entirely solve the problem of uninteresting actions being detected as interesting. False alarms in Figure 4.8 (top) will still be detected as false alarms in Figure 4.8 (bottom). The reasons for false alarms and how to mitigate against them using local image features are discussed later on.

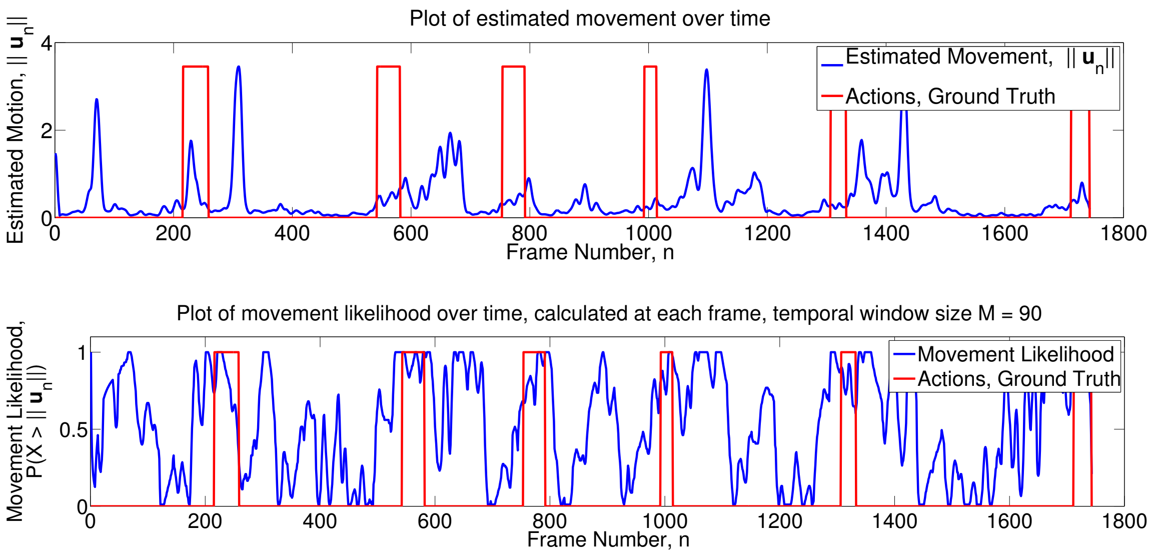


Figure 4.8: Before and after movement amount normalisation by calculating movement likelihood, top and bottom. By normalising the movement, every interesting action can be detected with a global threshold of $P(X \leq \|\mathbf{u}_n\|) \geq 0.7$. The movement profiles in the plots were smoothed for clarity, using a Gaussian kernel with $\sqrt{\sigma^2} = 3$.

alone would not result in perfectly accurate detection, as shown in Figures 4.7 & 4.8. It can be seen that many of the peaks that would be incorrectly detected as events occur directly after an actual interesting event. These particular movements are typically caused either by the athlete moving to set up the next shot, or someone else entering the shot. If the temporal window size M is high enough, these follow-on actions will not be regarded as significant, relative to the just-occurred action. However, many of these follow-on actions occur after the athlete has completed the action, and returned to a near rest state. For example, following the end of a golf stroke, the athlete will place the next ball on the tee. Many of these spurious actions can be mitigated by incorporating the prior knowledge that the duration of event is less than the duration between events, thereby adding a temporal constraint on the event detection system.

Temporal consistency is widely used in event detection and tracking [132, 91]. To intelligently add the above temporal constraint, this event detection system is now set up as a probabilistic state-machine. The proposed system is modelled by two states, $\mathbf{x} \in \{\text{“action”} = 1, \text{“not-action”} = 0\}$. In a probabilistic sense, the objective is to find the state at each frame n that maximises the posterior distribution $p(\mathbf{x}_n|\mathbf{y}_n)$, where $\mathbf{y}_n$ is the observed data at frame n. Using Bayes theorem to re-write the posterior distribution gives

$$p(\mathbf{x}_n|\mathbf{y}_n) \propto p(\mathbf{y}_n|\mathbf{x}_n)p(\mathbf{x}_n)$$

where $p(\mathbf{x}_n)$ is used to encode the prior knowledge of how likely each state is to occur, and the distribution $p(\mathbf{y}_n|\mathbf{x}_n)$ is the likelihood of observing the value $\mathbf{y}_n$, (i.e. $P(X \leq \|\mathbf{u}_n\|)$), assuming the system is in state $\mathbf{x}_n$. For this application, $p(\mathbf{y}_n|\text{“action”})$ is set to the calculated movement likelihood, and $p(\mathbf{y}_n|\text{“not-action”})$ is set to a constant threshold α,

$$\begin{aligned} p(\mathbf{y}_n|\mathbf{x}_n = 1) &= P(X \leq \|\mathbf{u}_n\|) \\ p(\mathbf{y}_n|\mathbf{x}_n = 0) &= \alpha \; . \end{aligned} \tag{4.4}$$

The prior distribution could then be set up such that $p(\text{“action”}) < p(\text{“not-action”})$, to encode the knowledge that at any given time it is more likely that an interesting event is not taking place. However, solving this simple modelling of the application for the most likely state sequence $\mathbf{x}$ at each frame n is effectively just a slightly complicated thresholding operation, and still does not incorporate any temporal constraints.

To introduce temporal consistency, the likelihood is modified from being per frame, to the sequence of states over the last M frames, i.e. $p(\mathbf{y}_{(n-M):n}|\mathbf{x}_{(n-M):n})$. Assuming the instantaneous likelihood $p(\mathbf{y}_n|\mathbf{x}_n)$ can be calculated independently at each frame, the new likelihood is

$$p(\mathbf{y}_{(n-M):n}|\mathbf{x}_{(n-M):n}) = \prod_{t=0}^{M} p(\mathbf{y}_{n-t}|\mathbf{x}_{n-t}) \; .$$

The likelihoods at each frame n, $p(\mathbf{y}_n|\text{“action”})$ and $p(\mathbf{y}_n|\text{“not-action”})$, are still given by the movement likelihood, $P(X \leq \|\mathbf{u}_n\|)$, and a constant threshold α respectively. To incorporate temporal constraints into the prior distribution, $p(\mathbf{x}_n)$, a first-order Markov chain is applied by a transition matrix $p(\mathbf{x}_{n-1}, \mathbf{x}_n)$, where $p(\mathbf{x}_{n-1}, \mathbf{x}_n)$ is the probability of transitioning between states $\mathbf{x}_{n-1}$ and $\mathbf{x}_n$ over consecutive frames.

From high-level knowledge of the coaching video, the time spent during an event is much less than the time between events. This is exploited to remove spurious follow-on events by discouraging changes in state to the “action” state. If the “action” and “not-action” states for a frame n are denoted by $\mathbf{x}_n = 1$ and $\mathbf{x}_n = 0$ respectively, the transition probability distribution is encoded as follows,

$p(\mathbf{x}_{n-1}, \mathbf{x}_n)$		$\mathbf{x}_n$	
		0	1
$\mathbf{x}_{n-1}$	0	z_1	$1 - z_1$
	1	z_2	$1 - z_2$

where z_1 and z_2 are variables controlling how likely it is to move from one state to the next. Typically z_1 and z_2 are set to values greater than 0.5 to encourage staying in the “not-action” state, and only changing to the “action” state when there's a high likelihood. Sensible values for z_1 to z_2 will be discussed later. The new likelihood and transition distributions are combined to give an updated posterior distribution

$$p(\mathbf{x}_{(n-M):n}, \mathbf{y}_{(n-M):n}) = \prod_{t=0}^{M} p(\mathbf{y}_{n-t}|\mathbf{x}_{n-t}) p(\mathbf{x}_{n-t-1}, \mathbf{x}_{n-t}) \; .$$

The posterior distribution is now of the typical problem form for solving with state-machine algorithms. Given enough training data, HMM's can be used to reliably and accurately estimate the most likely sequence of states [91, 94, 138]. However, the sport-type independent, real-time requirements of this application mean that training cannot be performed. Given the form of the posterior distribution, it is possible to use the Viterbi algorithm [55] to estimate the most likely state sequence. Consider calculating the negative logs of the likelihood and transition probabilities,

$$\begin{aligned}\lambda_l(\mathbf{x}_n, \mathbf{y}_n) &= -\log(p(\mathbf{y}_{n-t}|\mathbf{x}_{n-t})) \\ \lambda_p(\mathbf{x}_{n-1}, \mathbf{x}_n) &= -\log(p(\mathbf{x}_{n-t-1}, \mathbf{x}_{n-t}))\ .\end{aligned} \tag{4.5}$$

A cost for going from state $\mathbf{x}_{n-1}$ to $\mathbf{x}_n$ is then given by $\lambda_l(\mathbf{x}_n, \mathbf{y}_n) + \lambda_p(\mathbf{x}_{n-1}, \mathbf{x}_n)$. Consider then the total cost $\lambda(\mathbf{x}, \mathbf{y})$ for the last M frames of the sequence $\mathbf{x}$:

$$\begin{aligned}\lambda(\mathbf{x}, \mathbf{y}) = -\log(p(\mathbf{x}_{(n-M):n}, \mathbf{y}_{(n-M):n})) &= \sum_{t=0}^{M} -\log(p(\mathbf{y}_{n-t}|\mathbf{x}_{n-t})p(\mathbf{x}_{n-t-1}, \mathbf{x}_{n-t})) \\ \lambda(\mathbf{x}, \mathbf{y}) &= -\sum_{t=0}^{M} \log(p(\mathbf{y}_{n-t}|\mathbf{x}_{n-t}) - \sum_{t=0}^{M} \log(p(\mathbf{x}_{n-t-1}, \mathbf{x}_{n-t}))) \\ \lambda(\mathbf{x}, \mathbf{y}) &= \sum_{t=0}^{M} (\lambda_l(\mathbf{x}_{n-t}, \mathbf{y}_{n-t}) + \lambda_p(\mathbf{x}_{n-t-1}, \mathbf{x}_{n-t}))\ .\end{aligned}$$

The problem can now be set-up as a directed graph, where $\lambda_p(\mathbf{x}_{n-1}, \mathbf{x}_n)$ defines the edge costs, and $\lambda_l(\mathbf{x}_n, \mathbf{y}_n)$ the node (or state) costs. The most likely state sequence $\mathbf{x}$ is therefore the one that minimises the total sequence cost $\lambda(\mathbf{x}, \mathbf{y})$ along the path from the starting "node" at time $n - M$ (in this application it is assumed that the starting state $\mathbf{x}_{n-M}$ = "not-action"). Using a single forward pass of the Viterbi algorithm allows for on-line computation. Although it was found experimentally that at least one forward and backward pass improved parsing reliability, a single forward pass is sufficient for reliable parsing. The actual Viterbi algorithm used in this system is described in Algorithm 4.1.

An example is shown in Figure 4.9, with a comparison between actions detected by simply thresholding peaks in the smoothed movement profile, and those detected using the Viterbi algorithm shown in Figure 4.10. An advantage to using the Viterbi algorithm with only 2 possible states in the state machine model, is that it can be solved on-line and updated incrementally, making the most likely sequence very efficient to compute. The core components of the parsing system have been introduced. A discussion of the key component parameters is now presented.

4.5 Parameter Selection

This section discusses the five parameters with the greatest influence over system performance, listed as follows.

```
Input: $\lambda_l$, $\lambda_p$, start_cost, $M$, $n$
Output: $\mathbf{x}$, end_cost
$V(0,0)$ = start_cost(0);
$V(1,0)$ = start_cost(1);
for $i$=1 to $M$ do
    $V(0,i) = \lambda_l(\mathbf{y}_{n-M+i}|0) + \min(\ V(0,i-1) + \lambda_p(0,0),\ V(1,i-1) + \lambda_p(1,0)\ );$
    $V(1,i) = \lambda_l(\mathbf{y}_{n-M+i}|1) + \min(\ V(0,i-1) + \lambda_p(0,1),\ V(1,i-1) + \lambda_p(1,1)\ );$
    $\mathbf{x}(i) = \arg\min_j V(j,i);$
end
end_cost(0) = $V(0,M)$;
end_cost(1) = $V(1,M)$;
```

Algorithm 4.1: Single Forward pass Viterbi algorithm to find the most likely state sequence for the previous M frames from the current frame n. The variables λ_l and λ_p are those from Equation 4.5 respectively. The variable "start_cost" defines the cost of the initial state of the system, i.e. at frame $n - M$, and is usually set to the value of the variable "end_cost" output from the previous iteration of Viterbi algorithm. The resultant lowest-cost state sequence is returned in the output variable $\mathbf{x}$.

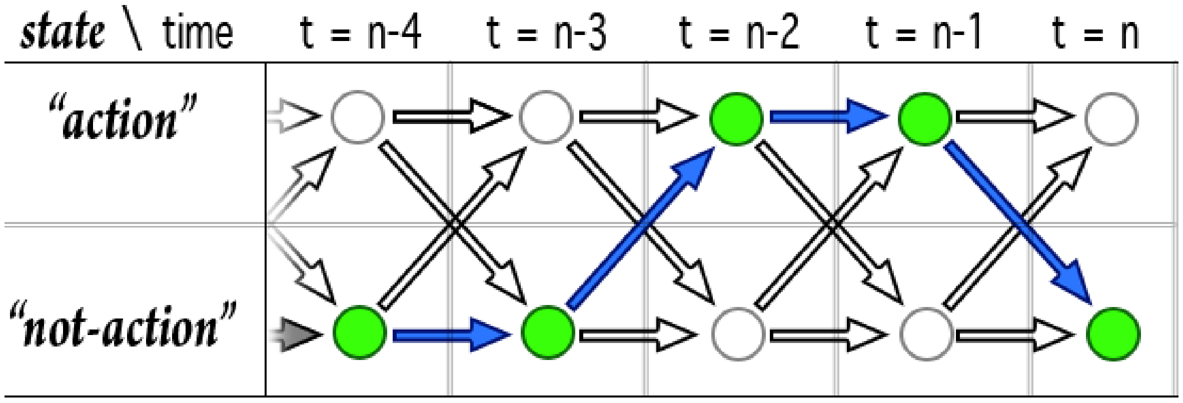


Figure 4.9: The Viterbi fully connected directed graph, or "trellis", used to find the most likely sequence of "action" or "not-action" states throughout the video. The trellis is constructed by taking the negative log of the likelihood and transition matrix probabilities, and using them as the state and edge costs, shown as the circles and arrows respectively. The most likely state sequence is then given by the path with lowest energy, from left to right, through the graph. An example path is shown in blue.

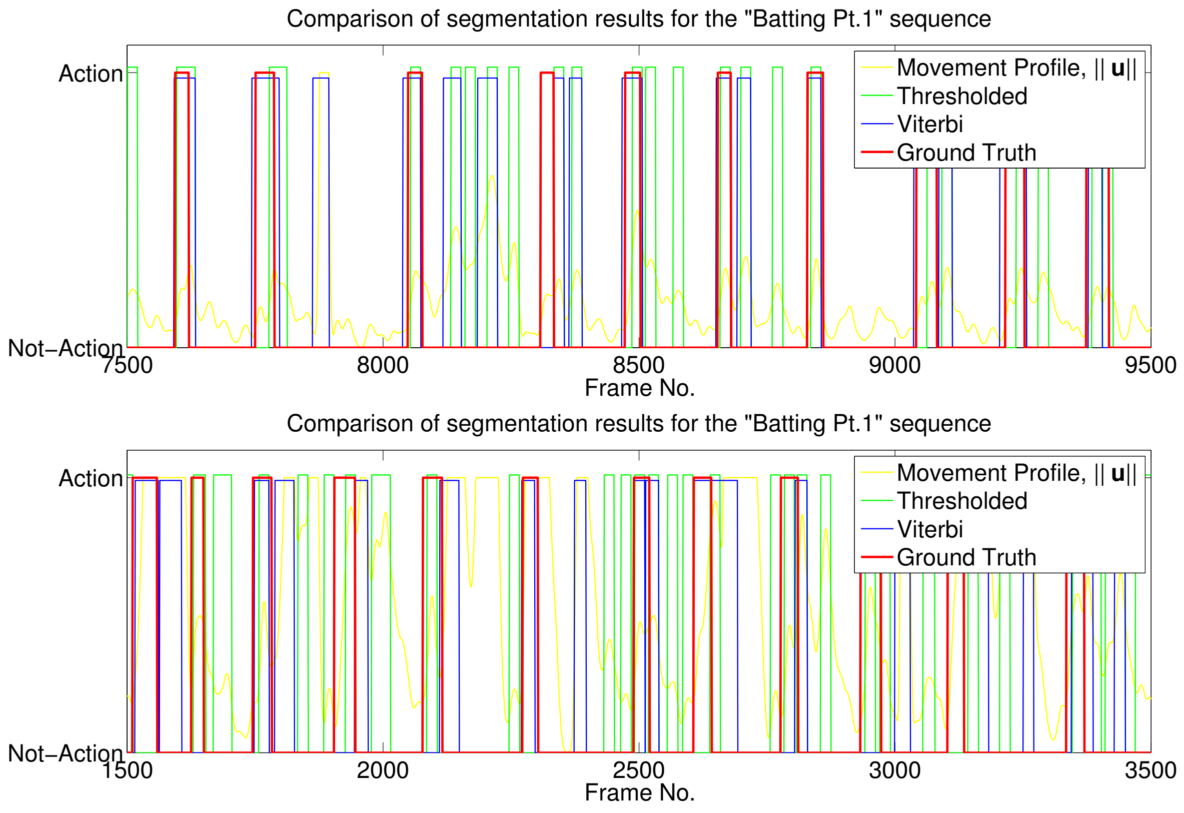


Figure 4.10: A visual comparison between events detected using the Viterbi algorithm, as described in Section 4.4, and simply thresholding the peaks of the movement likelihood p (red). The heights of the results have been staggered for clarity. Two difficult portions of the "Batting Pt.1" sequence were chosen for this comparison. The constant threshold α was set to give the best overall detection rate for this sequence, $\alpha = 0.2$ and 0.4 for "thresholded" and "Viterbi" respectively. Notice in the top plot, the thresholded method (green), performs reasonably well against the ground truth (yellow), capturing all interesting events and detecting 7 false alarms, compared to 6 false alarms using Viterbi (blue). In the bottom plot however, the thresholded method misses one interesting action, and detects 13 false alarms. The Viterbi method still detects all actions, and gives only 4 false alarms. It can be seen that the Viterbi will only detect an action if the likelihood of something interesting has been happening for a length of time. The comparison between simple thresholding and the Viterbi energy minimisation scheme is examined in more detail in Section 4.6.3.

1. N, the size of the temporal window used to mask the location of the athlete.
2. M, the size of the temporal window over which to calculate significance.
3. α, the fixed likelihood that an event is not happening in the current frame.
4. z_1, the probability of staying in the same "not-action" state between consecutive frames.
5. z_2, the probability of transitioning from an "action" state to "not-action" state between consecutive frames.

The size, N, of the temporal window used to mask the player is related to how long the user expects the player to stay in the same spot in the frame. In this system, a balance must be found that allows the system to adapt to the location of the current athlete quickly, but not so quickly that if the player moves somewhere else in the frame for a short moment the mask follows them. If the athlete is expected to stay in the same location, such as golf strokes or cricket bats, the size can be set to a reasonable value, such as $N \geq 500$ corresponding to about 20 seconds at 25 fps. On the other hand, if the athlete is likely to be moving around, for tennis rallies or cricket bowls, the user has two choices; to disable player masking, $N = 0$, and assume that all apparent movement in the frame is due to the athlete, or to set N extremely high, i.e. $N \geq 10,000$ (nearly 7 minutes at 25 fps), causing the system to calculate a very stable mask, encompassing all the probable player locations over a long time. Due to memory and real-time computation constraints, the latter setting is usually not preferred. In general, sensible values for most coaching sessions lie in the range $N \in [300, \ldots, 600]$, and $N = 0$ for sequences where player movement is expected to be seen throughout the image frame. The variable N may also be set by tracking the centre of the Gaussian mask over time, and increasing N adaptively if the motion of the mask is too high.

The window size, M, is the number of frames over which movement values are collected in $\mathbf{v}_n$ in order to estimate the significance of the current movement amount, $\|\mathbf{u}_n\|$, calculated at each frame to give the movement likelhood, $P(X \leq \|\mathbf{u}_n\|)$. In this work, the significance is defined as the probability of observing the current movement value or greater given the previously sampled data. This essentially allows us to identify local extrema in the movement signal $\|\mathbf{u}\|$ corresponding to interesting events. Experimentally, it has been found that good values for M are related to the durations of events within the sequence. This makes sense, as most actions in the coaching session are not exactly identical, and will exhibit different amounts of movement. If M is large enough to cover multiple actions, a single threshold for the likelihood is probably not going to reliably catch all of the actions. If M is too small, it is possible that frames between the actions with relatively low amounts of movement, such as background or small, irrelevant movements, will be flagged as "statistically significant". Although it will vary for each sequence, reasonable values for M include $M \in [20, \ldots, 100]$. It is possible to determine a good value for M given some ground truth, or even to ask the user for a fixed value. However, it is also possible to use the on-line parsing results to update values for M automatically, for example, by setting M to the mean length of the detected actions, providing it is within the above range of sensible values.

The variables α, z_1 and z_2 are used in the Viterbi energy minimisation scheme to apply temporal consistency to the detection. The variable α is the fixed likelihood that an action is not happening. α should therefore have values less than 1, to allow actions to be detected. In practice, it was found that a high, near 1 value for α results in good detection results. The variables z_1 and z_2 are the probabilities of transitioning from states "not-action" to "not-action", and "action" to "not-action" respectively. z_1 and z_2 encode the prior knowledge of how often actions occur relative to not-actions, for example, high values of z_1 encourage actions not to be detected unless their likelihood is very high, and high values of z_2 encourage sharp discontinuation of actions as soon as the (very high) likelihood drops. Conversely, low z_1 values cause more actions being detected, while low z_2 values will cause the durations of detected actions to become longer. As with the variable M, the values for z_1 and z_2 are related to the action durations and durations between actions. For the coaching sequences shown in this work, values for z_1 and z_2 that produced good detection results varied around $z_1, z_2 \in [0.6, \ldots, 0.9]$.

4.6 Experimental Results

The footage in these results is from actual sports coaching sessions, where athletes would like to be presented with an automatic summary of their actions. The sessions were filmed in typical conditions, and apart from resolution down-sampling, have not been edited or manipulated prior to processing. As a result, the sessions contain many real-world problems such as player occlusion, team-mates inadvertently walking across the camera shot, spurious background movement behind the foreground player, mid-shot camera position and focus adjustment, sudden shot cuts when the camera was unexpectedly switched off and on again, player movement during some *non*-interesting actions, and the absence of the player for long periods in the video.

Parameters for the system are selected for each session as described in Section 4.5. Each video is then processed off-line, resulting in the state sequence $\mathbf{x}$, where a run of $\mathbf{x}_n = 1$ indicates frames to be recorded as an interesting action. The results of the movement based video parsing system are evaluated first, both visually and numerically, in absence of the feature based false alarm detector. The numeric results give a statistical insight of the expected system performance for typical sequences, while the visual results show examples of segmented actions to prove both the measures used in the numerical analysis and the parsing system itself. Using the measures of performance described in the numeric analysis section, justification of the key components of the system is then provided. Following the results and design justification of the movement based parser, the proposed feature-based system for identifying incorrectly detected actions is then introduced and discussed.

4.6.1 Frame-based Numerical Results

To analyse the system numerically, a ground truth labelling, $\mathbf{g}$, is first manually extracted for each video, where $g_n = 1$ for an interesting action at frame n and $g_n = 0$ otherwise. The Precision and Recall measures typically found in Information Retrieval literature are used to evaluate the system,

$$\begin{aligned}\text{Precision} &= \frac{N_c}{N_c + N_f} \\ \text{Recall} &= \frac{N_c}{N_c + N_m}\end{aligned}$$

where N_c is the number of frames correctly identified as being "interesting" (i.e. where $x_n = g_n = 1$), N_f is the number of "false alarm" frames (where $x_n = 1$ and $g_n = 0$), and N_m is the number of missed frames ($x_n = 0$ and $g_n = 1$). Intuitively, a high Precision value means fewer uninteresting frames are detected, and a high Recall value means that fewer of the actually interesting frames are omitted in the "highlights" reel. An in-depth discussion of Precision and Recall with respect to video parsing is presented in Appendix A. Depending on the application requirements, different weights may be assigned to Precision and Recall values. For example a high Recall value might be preferred over a high Precision value if the user wants to correctly detect more action frames at the expense of accepting more not-action frames. For the purposes of these results, it is assumed that Precision is equally as important as Recall, and so the mean of the two error measurements, $\mu_{PR} = \frac{(\text{Precision} + \text{Recall})}{2}$, is used to rank the best results of the automatic parsing of the sequences, shown in the tables of Figure 4.11. The Precision and Recall values calculated between $\mathbf{x}$ and $\mathbf{g}$ give a good picture of how well the system is performing, however analysing Precision and Recall at this *frame*-based granularity is not intuitive. Dealing in numbers of correct events, instead of numbers of correct frames, is more meaningful.

4.6.2 Action-based Numerical Results

Consider that a detected event in $\mathbf{x}$ can be defined by a contiguous "plateau" of frames where $x_n = 1$ for all frames in the plateau. This is also true of the ground truth signal, $\mathbf{g}$. It is clear an event is correctly detected where two plateaus, in $\mathbf{x}$ and $\mathbf{g}$ respectively, share a sufficient overlap. Note that in contrast to the frame-based performance measure, a number of frames between the pair of plateau may not overlap entirely for an event to still be regarded as successfully detected. For the presented results and justification of decisions in system design, many thousands of parameter combinations are evaluated making it impossible to manually count the numbers of correct, false alarm and missed events for every detected event signal $\mathbf{x}$. Instead, an algorithm is required to automatically match plateaus between $\mathbf{x}$ and $\mathbf{g}$. Such an algorithm is not trivial, one such algorithm is developed and discussed in Appendix B. The proposed algorithm ensures an *injective* mapping between event plateaus in $\mathbf{x}$ and $\mathbf{g}$, that an event in $\mathbf{x}$ can map to at most one event in $\mathbf{g}$, and vice versa. Using the algorithm, the plateaus

Parameters					Frames				Precision & Recall		
N	M	α	z_1	z_2	N	t	N_f	N_m	P	R	μ_{PR}
150	**60**	**0.40**	**0.80**	**1.00**	**14940**	**16m 36s**	**7949**	**894**	**0.388**	**0.940**	**0.664**
500	60	0.40	0.80	1.00	14940	16m 36s	7979	886	0.387	0.941	0.664
500	250	0.20	0.80	0.80	14940	16m 36s	4166	2757	0.512	0.815	0.664
60	90	0.40	0.60	1.00	14940	16m 36s	9370	470	0.357	0.969	0.663
300	60	0.40	0.80	1.00	14940	16m 36s	8041	905	0.385	0.939	0.662

(a) "Bowling"

Parameters					Frames				Precision & Recall		
N	M	α	z_1	z_2	N	t	N_f	N_m	P	R	μ_{PR}
500	**35**	**1.00**	**0.60**	**0.80**	**14702**	**16m 20s**	**2184**	**3442**	**0.562**	**0.766**	**0.664**
500	35	0.60	0.80	0.80	14702	16m 20s	5063	1084	0.401	0.926	0.664
500	40	1.00	0.60	0.80	14702	16m 20s	2074	3603	0.571	0.755	0.663
300	35	0.60	0.80	0.80	14702	16m 20s	5135	1099	0.397	0.925	0.661
500	40	0.60	0.80	0.80	14702	16m 20s	4783	1355	0.410	0.908	0.659

(b) "Batting Pt.1"

Parameters					Frames				Precision & Recall		
N	M	α	z_1	z_2	N	t	N_f	N_m	P	R	μ_{PR}
2	**40**	**1.00**	**0.60**	**1.00**	**10165**	**11m 17s**	**5446**	**241**	**0.318**	**0.976**	**0.647**
2	35	1.00	0.60	1.00	10165	11m 17s	5810	211	0.305	0.979	0.642
2	40	0.40	0.80	0.60	10165	11m 17s	6066	231	0.295	0.977	0.636
2	35	0.60	0.80	0.80	10165	11m 17s	4441	820	0.350	0.919	0.635
2	40	0.60	0.80	0.80	10165	11m 17s	3949	1041	0.371	0.898	0.634

(c) "Batting Pt.2"

Parameters					Frames				Precision & Recall		
N	M	α	z_1	z_2	N	t	N_f	N_m	P	R	μ_{PR}
60	**20**	**0.80**	**0.80**	**1.00**	**6372**	**7m 4s**	**2175**	**362**	**0.449**	**0.943**	**0.696**
500	20	0.80	0.80	1.00	6372	7m 4s	2174	384	0.448	0.940	0.694
2	20	1.00	0.60	0.80	6372	7m 4s	1371	958	0.538	0.850	0.694
2	20	0.80	0.80	1.00	6372	7m 4s	1845	601	0.480	0.906	0.693
150	20	0.80	0.80	1.00	6372	7m 4s	2091	459	0.455	0.928	0.691

(d) "Batting Pt.3"

Figure 4.11: Results of automatic video parsing for interesting events against ground truth at the frame based granularity. Each sequence was parsed with a variety of parameters, and the top 5 parsing results with the highest mean Precision and Recall, μ_{PR}, values are presented. The best results in each sequence are highlighted in bold.

between $\mathbf{x}$ and $\mathbf{g}$ are sensibly matched, resulting in numbers of correctly detected actions N_c, false alarms N_f, and missed actions N_m. The Precision and Recall functions are redefined to measure at an *action*-granularity,

$$\begin{aligned}\text{Precision} = &\quad \frac{N_c}{N_c + N_f}\\ \text{Recall} = &\quad \frac{N_c}{N_c + N_m}\\ \mu_{PR} = &\quad \frac{(\text{Precision} + \text{Recall})}{2}\end{aligned}$$

where N_a is the actual number of actions present in the gound-truth, and N_c is the number of correctly detected actions. With the appropriate performance measures now defined, the results using action based granularity measures are presented in the tables of Figure 4.12.

The results in the tables of Figures 4.11 and 4.12 show reasonably high Mean Precision & Recall (MPR) values, indicating that the system performs well at automatically parsing the coaching videos used in the results. The results of both the frame-based and action-based results show higher values for Recall than Precision (for high MPR values). This means that overall there were more false detections than missed actions. This makes sense, as there is no way for the system to differentiate between changes in the movement profile induced by interesting player actions or other non-interesting events. For example, someone walking straight in front of the camera and into the player mask region, will produce an impulse-like effect in the apparent movement similar (but greater in magnitude) to that of a player action. Further examples of false alarms with their corresponding movement profiles are shown in Figure 4.14. Most of the false alarms in these sequences are due to people walking into the shot, the remaining false alarms are caused by unusual actions of the athlete, such as walking out of the shot or talking to team-mates. Some additional false alarms are due to spontaneous camera movement, panning and focus re-adjustment.

The high Recall values (> 0.95) for both action- and frame-based granularity indicate that nearly every action was correctly segmented. However, as previously discussed, a perfect Recall value of 1 can be achieved using the frame-based performance measure if the entire detected signal, $\mathbf{x}$. is 1. Using the action-based performance measure instead, the same detected signal $\mathbf{x}$ would give a very low Recall value. Therefore, the very high observed Recall values for the action-based measure indicate that the majority of interesting actions were not only correct detected, but correctly partitioned as well.

The action-based results for the "Batting Pt.1" and "Batting Pt.2" sequences in Figure 4.12 highlight two interesting effects the athlete can have on the results. Notice for the "Batting Pt.1" results, the Recall values are lower than observed for other sequences. Following inspection, it was noticed that the batsman was overly helpful to his team-mates, and would race to pick up and return the cricket ball to them. This caused two problems, the first was that the player mask was wider than it needed to be. The second problem was that high values of movement

Parameters					Actions				Precision & Recall		
N	M	α	z_1	z_2	N_a	N_d	N_f	N_m	P	R	μ_{PR}
500	**90**	**0.80**	**0.60**	**1.00**	**74**	**105**	**36**	**5**	**0.657**	**0.932**	**0.795**
300	90	0.80	0.60	1.00	74	106	37	5	0.651	0.932	0.792
500	90	0.60	0.80	1.00	74	103	35	6	0.660	0.919	0.790
500	90	0.40	0.80	0.80	74	110	41	5	0.627	0.932	0.780
300	90	0.40	0.80	0.80	74	112	43	5	0.616	0.932	0.774

(a) "Bowling"

Parameters					Actions				Precision & Recall		
N	M	α	z_1	z_2	N_a	N_d	N_f	N_m	P	R	μ_{PR}
2	**60**	**0.00**	**1.00**	**0.00**	**83**	**124**	**46**	**5**	**0.629**	**0.940**	**0.784**
300	40	0.80	0.80	1.00	83	103	31	11	0.699	0.867	0.783
500	40	1.00	0.60	0.80	83	101	30	12	0.703	0.855	0.779
150	40	0.80	0.80	1.00	83	108	35	10	0.676	0.880	0.778
500	35	1.00	0.60	0.80	83	105	33	11	0.686	0.867	0.777

(b) "Batting Pt.1"

Parameters					Actions				Precision & Recall		
N	M	α	z_1	z_2	N_a	N_d	N_f	N_m	P	R	μ_{PR}
2	**40**	**1.00**	**0.60**	**1.00**	**54**	**117**	**64**	**1**	**0.453**	**0.981**	**0.717**
2	40	0.00	1.00	0.00	54	126	72	0	0.429	1.000	0.714
2	35	0.60	0.80	0.80	54	113	61	2	0.460	0.963	0.712
60	40	0.80	0.80	1.00	54	90	42	6	0.533	0.889	0.711
60	40	0.60	0.60	0.20	54	130	76	0	0.415	1.000	0.708

(c) "Batting Pt.2"

Parameters					Actions				Precision & Recall		
N	M	α	z_1	z_2	N_a	N_d	N_f	N_m	P	R	μ_{PR}
60	**35**	**0.80**	**0.80**	**1.00**	**36**	**54**	**18**	**0**	**0.667**	**1.000**	**0.833**
2	20	1.00	0.60	0.80	36	52	17	1	0.673	0.972	0.823
300	20	0.40	0.80	0.40	36	40	9	5	0.775	0.861	0.818
2	35	0.80	0.80	1.00	36	50	16	2	0.680	0.944	0.812
2	20	0.80	0.80	1.00	36	54	19	1	0.648	0.972	0.810

(d) "Batting Pt.3"

Figure 4.12: Results of automatic parsing, given in terms of actions. Each sequence was parsed with a variety of parameters, and the top 5 parsing results with the lowest mean precision and recall error values are presented. The best results in each sequence are highlighted in bold.

(corresponding to the return throw) were sometimes present in $\mathbf{v}_n$, meaning a frame of an action with a high movement value which should have been marked interesting was not. Another player

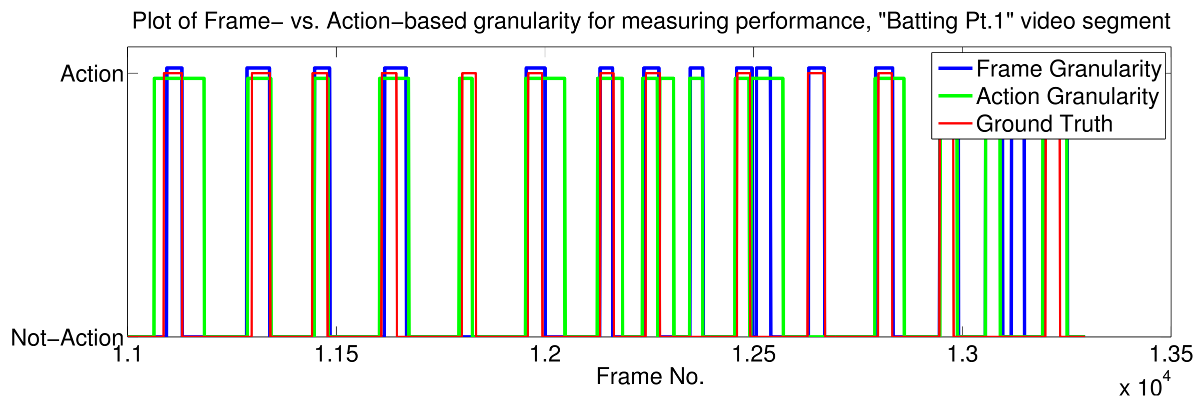


Figure 4.13: A comparison between frame- and action-based granularity in performance measurement for a segment of the "Batting Pt.1" sequence.

effect on the system is shown by lower than average Precision values for the "Batting Pt.2" sequence. It was discovered that while the batsman was eagerly awaiting the next bowl, the bat is quickly swung back and forth in situ before resting. This caused many mini-events to be detected before the actual interesting event.

Shown in Figure 4.13 are two detected action signals, $\mathbf{x}$ against ground truth for the "Batting Pt. 1" sequence. The detected signals are those with the highest MPR score using frame-based (blue) and action-based (red) performance measures, as shown in the tables of Figures 4.11 & 4.12 respectively. Although the accuracy is roughly the same (both miss one or two events, and both found one or two irrelevant events), notice the differences between detected actions for the two performance measures in Figure 4.13. As expected, the segmentation with the highest MPR for the frame-based measure has detected actions with boundaries that are closer to the ground truth. The parsing result with highest MPR for the action-based measure has detected action boundaries that often do not match up closely with that of the ground truth. However, these are probably visually more relevant to the user due to subjective ground truth error. As the action-based granularity allows some lee-way in the action boundaries between the detected and ground truth signals, the Precision and Recall values using the action-based performance measure will be higher.

4.6.3 Assessment of System Design

The presented system develops and extends upon previous relevant work in video parsing, as discussed in Section 4.1. However, the goals of this system are sufficiently dissimilar from any other work, that it makes comparisons with other implemented parsing systems difficult. For example, if an HMM system were to be trained on the movement likelihood data of the coaching video used in this work, it is expected that the HMM would perform at least as good or better. However, the need to train in advance means an HMM system would not be suitable for the real-

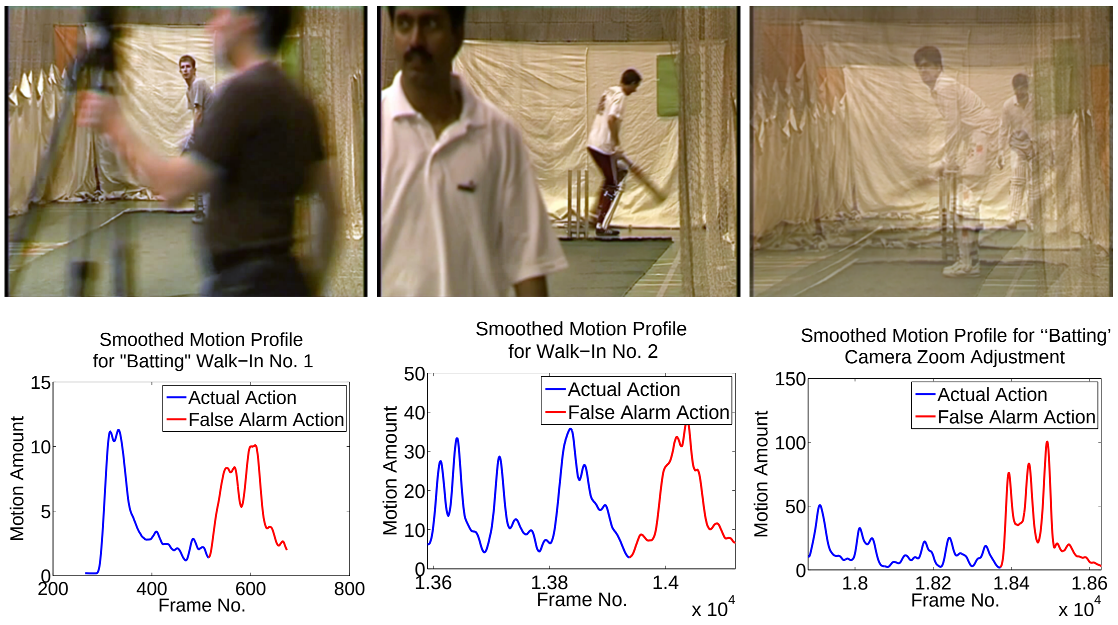


Figure 4.14: Examples of the most common causes for false detections sequences (top) and the associated movement patterns (bottom). Left and centre, people walking through the shots in the "Batting" sessions, and adjusting the camera while recording (zooming in and out in this case) shown in the right figure. Notice that in each case of false alarm detection, the erroneous movement profile (red) is indistinguishable from correct movement profiles (blue). The motion amount profiles have been smoothed by a Gaussian filter of $\sigma = 5$ for clarity.

world application. Related work in motion detection would be more applicable to compare the presented system, however it would be unfair (and misleading) to compare a motion detection algorithm that is more suited to surveillance footage than sports coaching video. Instead of comparing against other algorithms, the design of this system itself is evaluated, illustrating the effects of the various system components.

As the proposed method of calculating numbers of correctly detected actions allows the automatic evaluation of thousands of possible parameter combinations, it is possible to statistically evaluate and justify the effects of the various system design choices. For example, the choice to temporally mask the player throughout the video seems like a reasonable idea, but how well does it affect the performance of the system overall? The following experiment aims to find the effect of each component by comparing the system performance with the component, then "switching off" the component, measuring the performance again, and comparing the difference between the two.

The example component of the temporal player mask will be used to explain the proposed system for justifying system components. First, thousands of combinations of parameters are generated, using the sensible ranges discussed in Section 4.5. Another set of thousands of parameters is then permuted, this time with the temporal mask parameter, N, set to 0, effectively switching off the temporal mask. For each parameter combination in both sets, the system processes the video, and calculates the Mean Precision and Recall (MPR) values. The MPR distributions between the $N \neq 0$ and $N = 0$ parameter sets are then analysed for any apparent differences in performance.

There are a number of ways to compare populations of two distributions. In the presented assessment, the two sample non-parametric, distribution-shape agnostic *Kolmogorov-Smirnov* (K.S.) test statistic is proposed to compare the proportions of the "best" $N = 0$ and $N \neq 0$ distributions. Reasons for using the K.S. statistic in this situation are presented in Appendix C. The data being compared will be limited to the 50 samples with the highest MPR values from each of the $N \neq 0$ and $N = 0$ parameter sets. Comparing the best of one distribution with the best of the other makes sense, as it is expected that the user will choose sensible parameters that result in values within the 50 best MPR values.

The statistical test to be undertaken is formally defined as follows. The null hypothesis to be tested, H_0, is that the highest 50 MPR values of the distribution $N \neq 0$ are less than or equal to those from the $N = 0$ distribution. The alternative hypothesis is that the values of the distribution $N \neq 0$ are greater than those of the $N = 0$ distribution. The test statistic used is the two-sample K.S. statistic, $D_{n,n'}$, and will be tested at the significance level α. Details and examples of the K.S. statistic is discussed further in Appendix C. In short, the two-sample K.S. test statistic measures the maximum divergence between the pair of cumulative frequencies of each of the two distributions. A value of $D_{n,n'} = 0$ in this case implies that none of the samples in the $N \neq 0$ population are greater than the $N = 0$ samples (indicating the component provides no significant improvement in the system), a value of $D_{n,n'} = 1$ implies that all of the $N \neq 0$ samples are greater than all of the $N = 0$ samples (the component probably provides a significant performance improvement), and a value of $D_{n,n'}$ between 0 and 1 exclusive implies some overlap in the two pdfs (the performance change may or may not be significant depending on the degree of overlap). These hypotheses tests are performed on the three major components of the system to justify their roles. The three parts in question are:

1. The adaptive mask that isolates the athlete in the frame, controlled by the parameter N. The mask is "switched off" by setting $N = 0$.

2. The "movement significance" window that gives an idea of how significant the current movement magnitude is relative to previous movement values, the size of which is given by M. Again, the use of this temporal window is disabled by setting $M = 0$.

3. The temporal consistency constraint applied by the Viterbi energy minimisation scheme, denoted by α. To test the system without the temporal consistency (denoted by $\alpha = \emptyset$), the simple thresholding scheme given by Equation 4.5 is used instead.

The results of the tests for significance are presented in the Table 4.1. A number of interesting things to note arise from these results. Firstly, 11 of the 12 sequences have rejected the null

Sequence	*Parameter*	$D_{n,n'}$	p	Reject H_0?
"Bowling"	N	0.46	0.000	Reject
"Batting Pt.1"	N	0.74	0.000	Reject
"Batting Pt.2"	N	0.66	0.000	Reject
"Batting Pt.3"	N	0.70	0.000	Reject
"Bowling"	M	0.00	1.000	Accept
"Batting Pt.1"	M	1.00	0.000	Reject
"Batting Pt.2"	M	1.00	0.000	Reject
"Batting Pt.3"	M	1.00	0.000	Reject
"Bowling"	α	1.00	0.000	Reject
"Batting Pt.1"	α	1.00	0.000	Reject
"Batting Pt.2"	α	1.00	0.000	Reject
"Batting Pt.3"	α	1.00	0.000	Reject

Table 4.1: Table of results for statistical significance of the three major system components.

hypothesis, indicating that each of the design choices provides a significant improvement to system performance. The caveat of these tests is that the conclusions are valid for these sequences only. However, it is expected that given similar coaching video scenarios, results comparable to these would be observed.

Another interesting thing to note is that the test statistics, $D_{n,n'}$, are generally very high, between 0.46 and 1. Intuitively, a value of $D_{n,n'} = 1$ implies that all the samples from omitted system component, i.e. $N = 0$ for the player mask, had lower MPR values than with the component included. The value of $D_{n,n'} = 0$ for the $M = 0$ result on the "Bowling" sequence shown in Table 4.1, means that all of the MPR values in the $M = 0$ distribution were higher than or equal to values in the $M \neq 0$ "best selection" distribution. In this case of the "Bowling" sequence, the "significance window" size M was shown to have no significant effect on performance. This is most likely due to the the large image area occupied by the cricket bowlers causing a clearer than usual separation between movements of interesting actions and not-action movements.

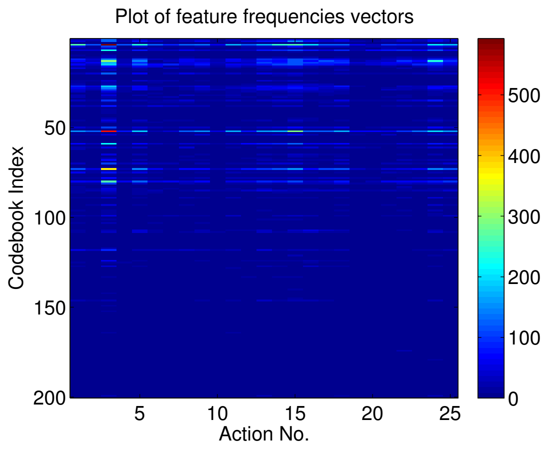
Lastly, the p values appear to be effectively binary. Ordinarily this would indicate something is wrong, such as incorrectly assuming a distribution belongs to a particular family. In this case, the p value is either 0 or 1 as the statistical *power* of the K.S. test becomes very high with a reasonable number of samples per distribution (Further discussion on the power of the K.S. test is given in Appendix C). In this test the number of "best selection" samples, $n = n' = 50$, was chosen specifically to be high enough to provide a high statistical power, yet low enough such that each of the 50 best parsing results are actually relevant and of a satisfactory quality to the user. As previously noted, the values of the test statistic, $D_{n,n'}$ are generally very high, which combined with a very high test power, results in binary p values.

4.7 Using Local Image Features to Detect False Alarms

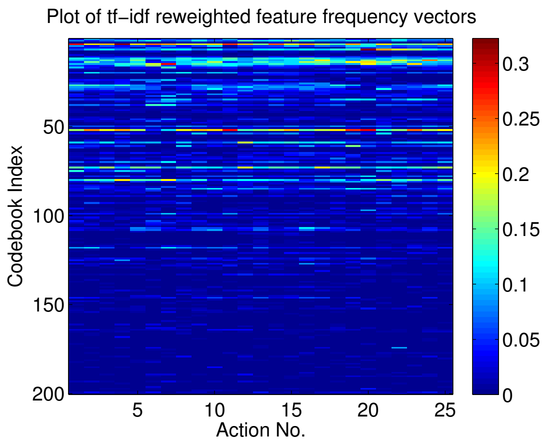
The main issue with the presented system so far is the reasonably high rate of false alarms, as shown by the comparatively low Precision values in the results of both the frame (Table 4.11) and action (Table 4.12) granularities. In practice, while watching a summary video (highlights reel) of the shots one after another, it is sometimes difficult to notice when a false alarm has occurred. Many of the false alarms are introduced by other subjects walking in front of the camera, or the athlete leaving and subsequently returning into the shot. Occasionally other high motion actions will trigger a false event detection, such as the throwing of a ball instead of batting it. As the user typically scans the video for just the interesting shots, the false alarms are generally ignored. However, if the system is being used to browse for particular shots, either in real-time or offline, the walk-in false alarms are distracting. For instance, as a still image thumbnail of the shot does not always show the walk-in, the clip must be viewed before its actual relevance is known. Using the framework of the presented system, there is no clear way to reduce the rate of false alarm detection. An additional step using local image features is proposed to better detect these false alarms.

The high Recall values Tables 4.11 & 4.12 show that the majority of actions are successfully detected, it is therefore sufficient to keep the existing system and apply a post detection step to remove false alarms. This false alarm detection is set up as a shot clustering exercise, where the actions of the video is classified as either relevant or false alarm. Recall as well that an athlete region-of-interest mask exists. Given that an action is only signalled by movement within this region, it is expected that the athlete should be present in the region for most of shot. Using this knowledge, it should be possible then to model the athlete somehow, and detect departures from this model as false alarms. For example, if the athlete and background have different colour distributions, it should be possible to detect if the athlete has left or walked into the interest region at some point during the shot by looking at the colour distributions of the shot. As discussed earlier in Section 4.1, Lu & Tan [106] use colour histograms from the entire shot to automatically group events. Colour histograms are simple to compute, and in theory should help distinguish between the colour of the athlete and that of a walk-in. However, the quality of the colour space can not be guaranteed, for example video compression often quantises the chrominance channels (i.e. Cb & Cr of YCbCr, and H & S of the HSV colourspace). In the work of Lu & Tan, the shots being classified usually contained a large degree of colour variation, allowing easier discrimination between shots of other classes. In the presented system, the colour differences between a good shot and a false alarm are minimal. Combined with the quantisation problem, colour is not a very discriminative feature for detecting false alarms. Instead, a method based on comparing image content in the shots is proposed.

In Chapter 2, the “bag-of-words” model was applied to feature point “visual words” for object detection and classification applications. It is now repurposed to allow clustering of shots based on their image content. The idea is to model the image data in the detected actions



(a) Before Reweighting

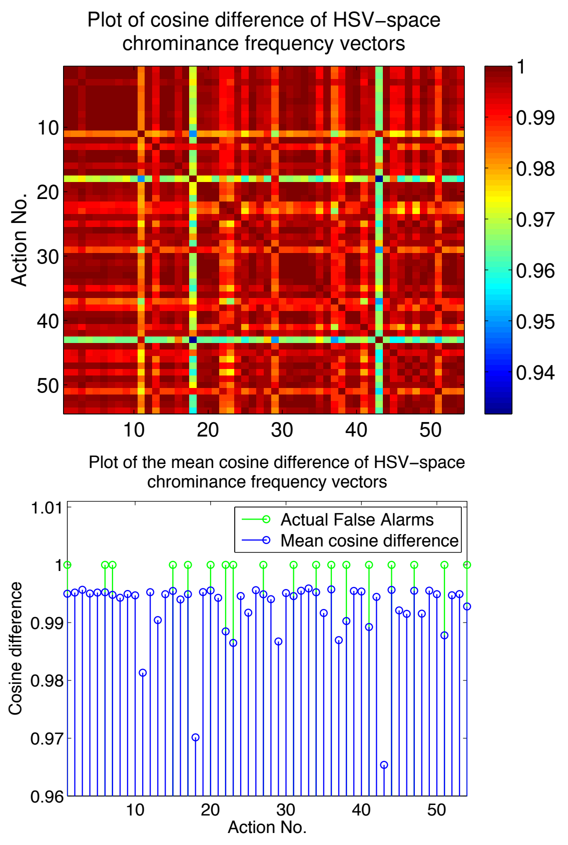


(b) After tf-idf Reweighting

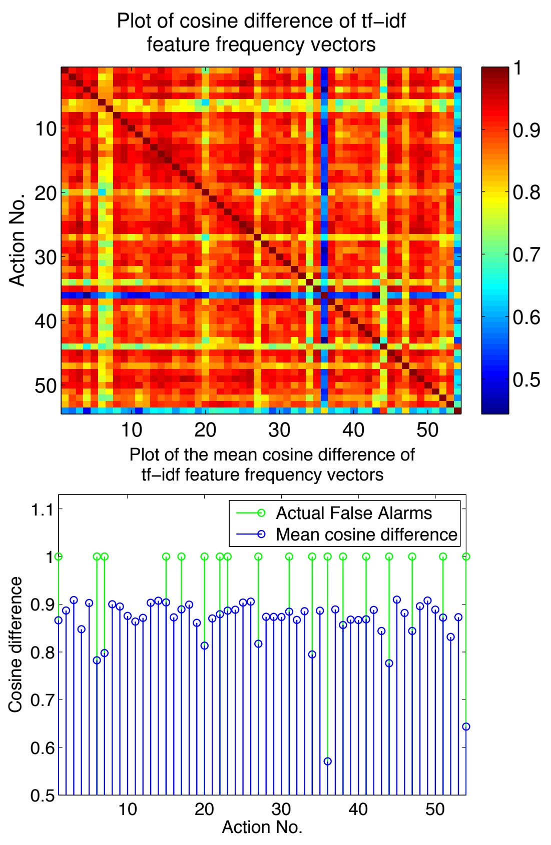
Figure 4.15: Examples of feature frequency vectors of detected actions in the "Batting Pt.3" sequence, before (**v**) and after tf-idf re-weighting (**t**). Only the first 25 actions are shown for clarity. Popular features tend to belong to lower code-book indices as a product of the Mean-Shift algorithm. Notice that in the plot on the left, actions are generally represented by only a few, very popular feature indices. Visually, the improvement in tf-idf re-weighting can be seen by the change in contrast between the two plots, where features very popular throughout the sequence are weighted down, and those popular within the action are weighted higher for that action.

by the populations of image features in the action frames. In the work of Sivic & Zisserman [160], features are first calculated throughout the entire video before being quantised to a large ($> 50,000$ word) vocabulary. This is extremely computationally expensive, and so does not fit well with the record & review application objectives. However, it is possible to exploit the constrained settings of the presented system to offer a similar more limited system. For example, instead of using a large vocabulary to capture the maximum amount of variation throughout the video, it is known in advance that the camera will generally be fixed, the athlete is usually in same throughout the video, and will generally be found only in the region of interest mask. Ultimately, this means that the scene content will change very little throughout the video, compared to the cinematic films used in Sivic & Zisserman. This allows two important things; features only need to be calculated in the region of interest mask, and the code-book vocabulary for quantisation can be very low (200 - 300) and still manage to capture the scene content well.

The offline case is now described, with the on-line case discussed at the end. Consider that S is the set of all detected actions in a sequence, where $s = \{a, \dots, b\}$, $s \in S$ is one such action comprised of frames from a to b. The parameters of the (Harris-Laplace) feature detector are set to return between 100 and 200 features for the image region inside the athlete mask. Approximately $5,000 - 10,000$ feature descriptors are used to build the code-book from features calculated on random action frames. The descriptors are clustered using the Path Assigned Mean-Shift [134], which provides a significant speed-up over traditional mean-shift approaches.



(a) HSV



(b) tf-idf features

Figure 4.16: Comparison of colour (left col.) vs. tf-idf feature (right col.) frequency vectors for discrimination of false alarms in the 54 detected actions of the "Batting Pt.3" sequence.

Following the feature detection on an image, the detected features are now represented by the code-book indices of their nearest clusters in the descriptor space.

In their bag-of-words model, Sivic & Zisserman [160] define a "document" vector (before tf-idf weighting) as the frequencies of feature points calculated on a single key-frame. Instead, the frequencies of features from all the frames belonging to an action s are used,

$$\begin{aligned} \mathbf{v}_s &= [v_{1,s}, \ldots, v_{i,s}, \ldots, v_{N_c,s}] \\ v_{i,s} &= \textstyle\sum_{f \in s} \sum_{d \in D_f} [q(d) = i] \end{aligned} \tag{4.6}$$

where D_f is the set of feature descriptors in frame f, q is the quantisation function that maps a feature descriptor d to its code-book index, $[\phi]$ is the logical operator which is 1 if the predicate ϕ is true and 0 otherwise, and N_c is the number of indices in the code-book. The vectors $\mathbf{v}_s$ are calculated for every detected action $s \in S$. The similarity between a pair of actions can be given by the cosine difference between a pair of vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. However, as the content is expected to very similar throughout, relatively few unique features are produced (hence the low number of code-book clusters), resulting in some features with extremely high occurrence frequencies

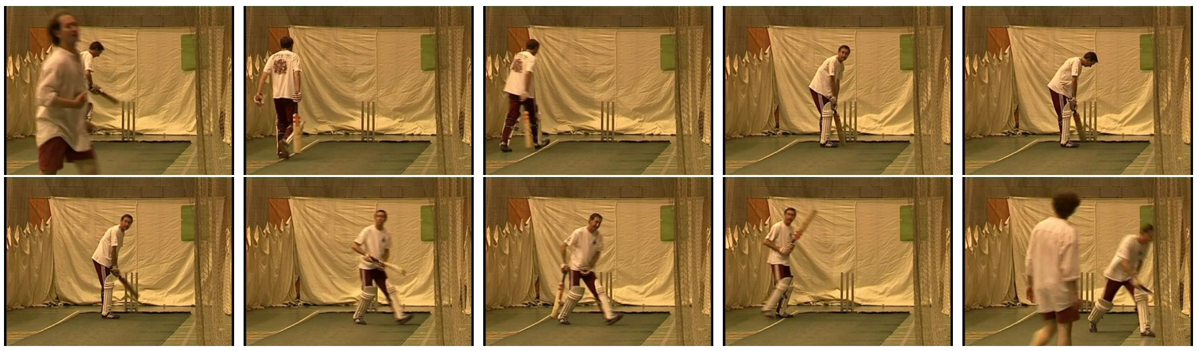


Figure 4.17: Examples of correctly identified false alarms (top row), and missed false alarms (bottom row) for the "Batting Pt.3" sequence. The relevant actions numbers, the mean cosine differences of which can be seen in Figure 4.16, for the correct top row are 6, 7, 17, 34 & 36. While the missed actions of the bottom row belong to actions are 1, 15, 17, 20 & 22.

($>$ 20% of the feature population). To normalise the frequency vectors relative to the per-action ("tf") and global ("idf") feature populations the tf-idf re-weighting scheme is used,

$$\begin{aligned} \mathbf{t_s} &= \{t_{1,s}, \ldots, t_{N_c,s}\} \\ t_{i,s} &= \frac{v_{i,s}}{\sum_{i\in[1,\ldots,N_c]} v_{i,s}} \log\left(\frac{\sum_{a\in S}\sum_{i\in[1,\ldots,N_c]} v_{i,a}}{\sum_{a\in S} v_{i,a}}\right). \end{aligned}$$

An example showing the effects of applying the tf-idf re-weighting are shown in Figure 4.15. Using the cosine difference measure, one action can be compared to another for similarity in image content. To help illustrate the effectiveness of using tf-idf feature frequency vectors (versus colour histograms), a simple analysis using cosine distance matrices is shown in Figure 4.16. In this example, the colour feature vector is given by the $10 \times 10 = 100$ bins of the 2D histogram of Hue and Saturation values (from HSV colourspace) from all the frames in the action. The tf-idf vector is calculated from Equation 4.7. In Fig. 4.16 (top row), cosine distance matrices for colour and tf-idf feature vectors belonging to the 54 detected actions (i.e. for tf-idf, distance matrix entry $d(i,j) = \frac{\mathbf{t}_i \cdot \mathbf{t}_j}{\|\mathbf{t}_i\|\|\mathbf{t}_j\|}$ $\forall (i,j) \in [1,\ldots,54] \times [1,\ldots,54]$). Horizontal or vertical lines with relatively low values indicate strong disagreement with other actions. In Fig. 4.16 (bottom row), the average distances for a given action (blue) vs. actions known to be false alarms (green). In the colour plot (left), no apparent correlation exists between the mean cosine difference and false alarm actions. However, in the tf-idf plot, there is a clear relationship, for example, a simple threshold of 0.85 is sufficient to detect half of the false alarms. Even in this simple experiment, local image features are far better at identifying false alarm actions than colour.

The visual analysis of the distance matrices in Figure 4.15 is interesting, however it is used simply to demonstrate the clarity with which local image features can identify false alarms using tf-idf frequency vectors. To use the vectors more powerfully, the set of frequency vectors $\mathbf{t}$ are k-means clustered, using the cosine distance metric, and setting $k = 2$ corresponding to the number of expected action classes: correct or false alarm. To ensure that there are actually false alarms present, the cosine difference between the two cluster centroids is first calculated. If the distance is sufficiently great (i.e. the cosine distance is less than 0.7), it is assumed that there are false alarms in the detected actions. Assuming that there are more correctly detected actions than false alarms, the cluster centroid with the fewest assigned action vectors, $\mathbf{t}_i$, is designated the "false alarm" cluster[4]. The detected false alarm actions (corresponding to the actions assigned to the false alarm cluster) are then rejected. Examples of correctly identified, and missed, false alarms are shown in Figure 4.17. From the figure it is seen that many of the good false alarm detections (top row) belong to walk-ins of the athlete (figs. 2 & 3) or other team-mates (fig. 1), with the remainder belonging to other uninteresting actions, such as setting up (figs. 4&5).

The proposed detection method is now applied to the various sequences using the parameter settings with the highest MPR values of Figure 4.12, the results are shown in the Table below.

		Without False Alarm Detection						With False Alarm Detection					
		Actions			Prec. & Recall			Actions			Prec. & Recall		
Sequences	N_a	N_d	N_f	N_m	P	R	μ_{PR}	N_d	N_f	N_m	P	R	μ_{PR}
Bowling	74	105	36	5	0.66	0.93	0.8	**74**	**5**	**5**	**0.93**	**0.93**	**0.93**
Batting Pt.1	83	124	46	5	0.63	0.94	0.78	**98**	**20**	**5**	**0.80**	**0.94**	**0.87**
Batting Pt.2	54	117	64	1	0.45	0.98	0.72	**64**	**11**	**1**	**0.83**	**0.98**	**0.90**
Batting Pt.3	36	54	18	0	0.67	1	0.83	**44**	**8**	**0**	**0.82**	**1.00**	**0.91**

Table 4.2: Results of applying the feature-based false alarm detector to the detection results of Figure 4.12.

The results show a marked improvement in false alarm detection rates. Also, as this classification technique has the ability to detect interesting actions as false alarms (i.e. false alarm of a false alarm), the number of missed actions N_m can potentially be affected. The question now is how feasible is it to perform this process online. To summarise the operations, the false alarm detector needs to calculate features, perform a reasonable sized clustering task for code-book generation, quantise the calculated descriptors to create action feature histograms, re-weight the tf-idf histograms followed by another small clustering task. From some initial experiments performed, the feature calculation and clustering appear to be the major performance bottlenecks. As the area in which the features to be calculated is limited by the player mask, it is possible to get near-real time feature detection without any modifications or quality sacrifices. In order to get enough data, the code-book generation requires that features have been detected on a large number of detected action frames already. Clustering using the PAMS algorithm usually takes between 2-3 minutes depending on the number of size of the data to be clustered. One possible method for online use is to calculate features on the detected actions, skipping some frames in

[4]This of course will fail if the number of correct actions is less than the number of false alarms, but at that stage something has probably gone wrong with the coaching session.

order to stay real-time. A FIFO buffer could then be used to keep track of the most recently detected features. Periodically, a new code-book is then generated based on the accumulated data in the buffer. False alarms can then be detected in the incoming actions using the new code-book. Given that the scene content of the coaching footage generally does not vary much in the short term, some form of incremental codebook algorithm would be very useful for making the online version of this false detection system more elegant.

4.7.1 Visual Results

To show examples of what would actually be presented to the user, interesting actions segmented from various sequences are shown in the accompanying DVD, detailed in Appendix D.1. Firstly, the parsing results with the highest frame-based MPR granularity (from Figure 4.11) are shown for each sequence. Notice that although the "core" action frames are generally correctly captured, the disagreement between the detected signal and ground truth at the beginning and end of action boundaries significantly lowers the Precision score. Next, highlights videos compiled by the systems using parameters with the highest action based MPR values are presented. Two sets of videos show the (before and after) results of applying the feature based false alarm detection stage (i.e. the systems shown in Figure 4.2).

Further visual inspection of the segmented signals, $\mathbf{x}$, from the sequences are shown in Figure 4.18. Examples of frames from whole, correctly segmented actions are shown in Figure 4.19. The presented frames are from actions that can be seen in the plots of Figure 4.18. The "Bowling" action is from the segment between frames 12,950 to 13,040, taken 10 frames apart.. The "Batting pt. 1" action is between frames 11,280 and 11,340, taken approximately 7 frames apart. The "Batting pt. 2" action is from frames 6,671 to 6,707, taken approximately 4 frames apart. The "Batting pt. 2" action is between frames 3,511 to 3,553, taken approximately 5 frames apart.

Examples of frames from false alarms in the "Bowling" and "Batting pt. 1" sequences, and undetected actions in the "Batting pt. 2" and "Batting pt. 3" sequences are shown in Figure 4.20. Again these actions correspond to actions seen in the plots of Figure 4.18. The relevant frame ranges for the actions are 13,260 to 13,320, 10,530 to 10,570, 6,892 to 6,926 and 4,491 to 4,534 for the "Bowling", "Batting pt. 1", "Batting pt. 2" and "Batting pt. 3" respectively.

4.8 Discussion

This chapter presented a novel system for automatically detecting interesting events in sports coaching video without any prior knowledge. It was shown that sports actions can be reliably parsed using low-level movement cues, simple player masking and a computationally efficient scheme for enforcing temporal consistency. The various design choices of the motion parsing system were statistically justified, and results were calculated at two semantically different levels;

frame and action. A simple content analysis technique using local image features was then used to significantly reduce the number of incorrectly detected actions of the motion based parsing system.

Ultimately, the system can be viewed as an elaborate motion detector, that exploits "domain specific knowledge" of coaching video. The detection criteria are tuned to the expected, repetitive, "ready-then-action" movements. The expected single athlete of the one-on-one coaching session allows a simple player mask region to be established. Motion detection has been explored at length, and a great deal of work is focused on the analysis of sports video, however no-one has looked at a generic, "start-stop" motion parser for sports footage.

The results shown in tables of Figures 4.11 and 4.12 show reasonably high mean Precision and Recall values indicating the system performs reasonably well. This is confirmed by the visual results shown in Section 4.7.1 and by the parsed videos on the accompanying DVD, detailed in Appendix D.1.

Even with the feature based detection step, one of the main problems with the current system design is the number of false positives. These incorrect actions are usually the result of other subjects entering the frame, or of the athlete moving in a way that appears to be "interesting", such as returning a ball or lining up the next action. From a practical point of view, it is far easier for the user to spot which events are not interesting (and to delete them) once they have been parsed, than to manually mark the beginning and end of each event. The approx. 10% false alarm rate is therefore within most users acceptable "manual effort" tolerance.

In the movement parsing system (before using local image features), the player mask region is the only system component that attempts to mitigate false positives. While the mask does lower the number of incorrectly detected actions, it currently only allows for one athlete to be on the screen at any time. This was evident in the "Bowling" video, where many bowlers would enter the frame suddenly from either the left or right hand side of the frame. A system for allowing multiple subjects to be tracked could help provide more flexible active player regions than a single, simple Gaussian weighted ellipse. However, using the current system on the "Bowling" video, it was found that if the temporal mask window size N was set high enough (> 300), the Gaussian ellipse of the player region grew large enough to encompass the entire image frame, and the system continued on as usual. In cases such as this, where multiple athletes are consistently present (instead of "walk-in's"), the player mask "fails gracefully", allowing the system to function well.

The simple technique of using weighted histograms of feature codebook indices to represent and match the image content of the detected actions allowed significant reduction of falsely detected actions. The success of this secondary detection step lies in the highly constrained scenario of the sports coaching session, effectively reducing the possible range of image content. In turn, the number of visual words in the dictionary is reduced by several orders of magnitude, allowing a feasible codebook computation to be performed in near real-time. Additionally, as the content is expected to be very similar within each action, it is vitally important that the most relevant and descriptive features are weighted higher. The success of the feature based false alarm detector has demonstrated the power of the tf-idf feature weighting for use in content analysis.

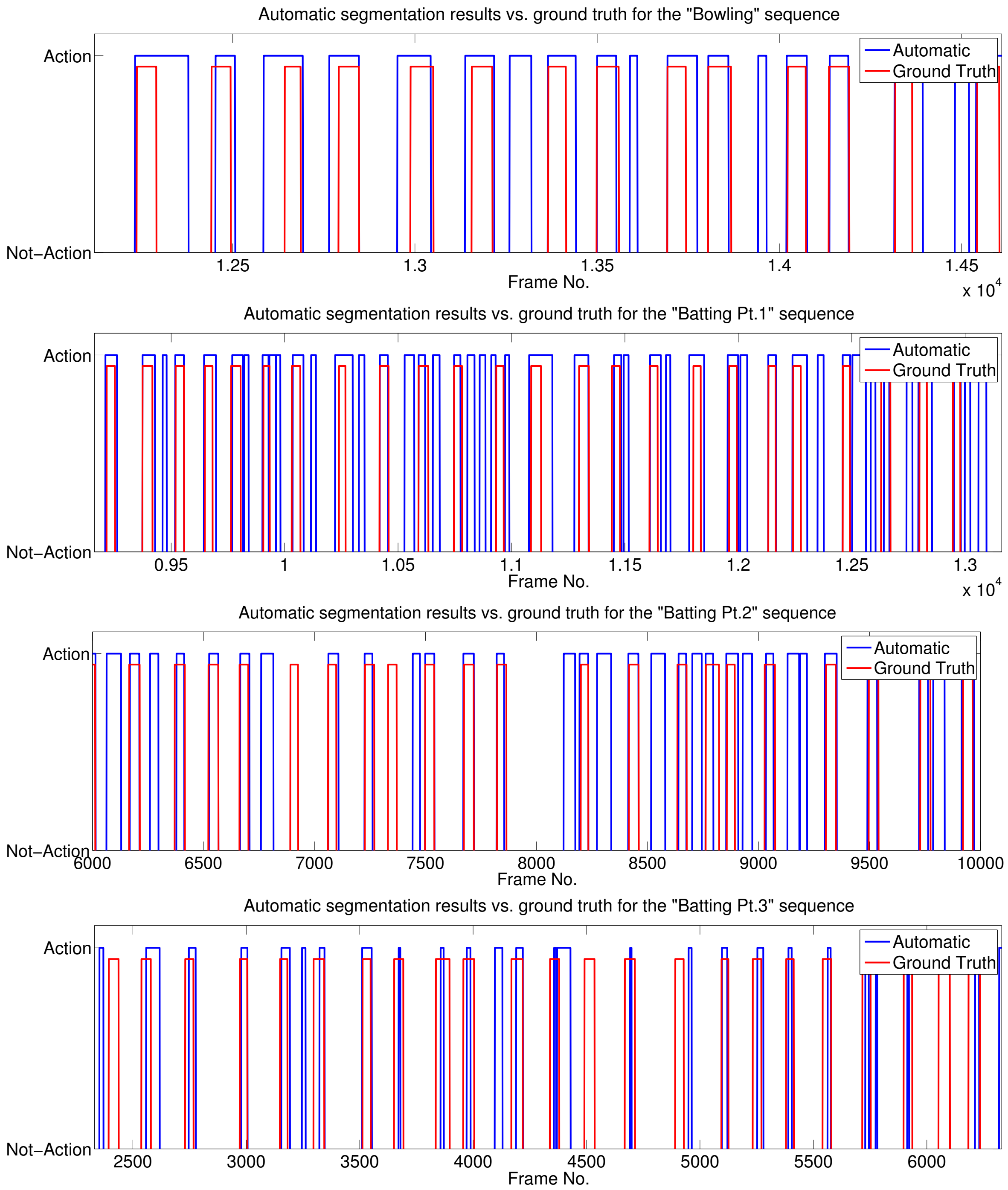


Figure 4.18: Examples of actions automatically parsed for a number of video sequences. The heights of the results have been adjusted for clarity.

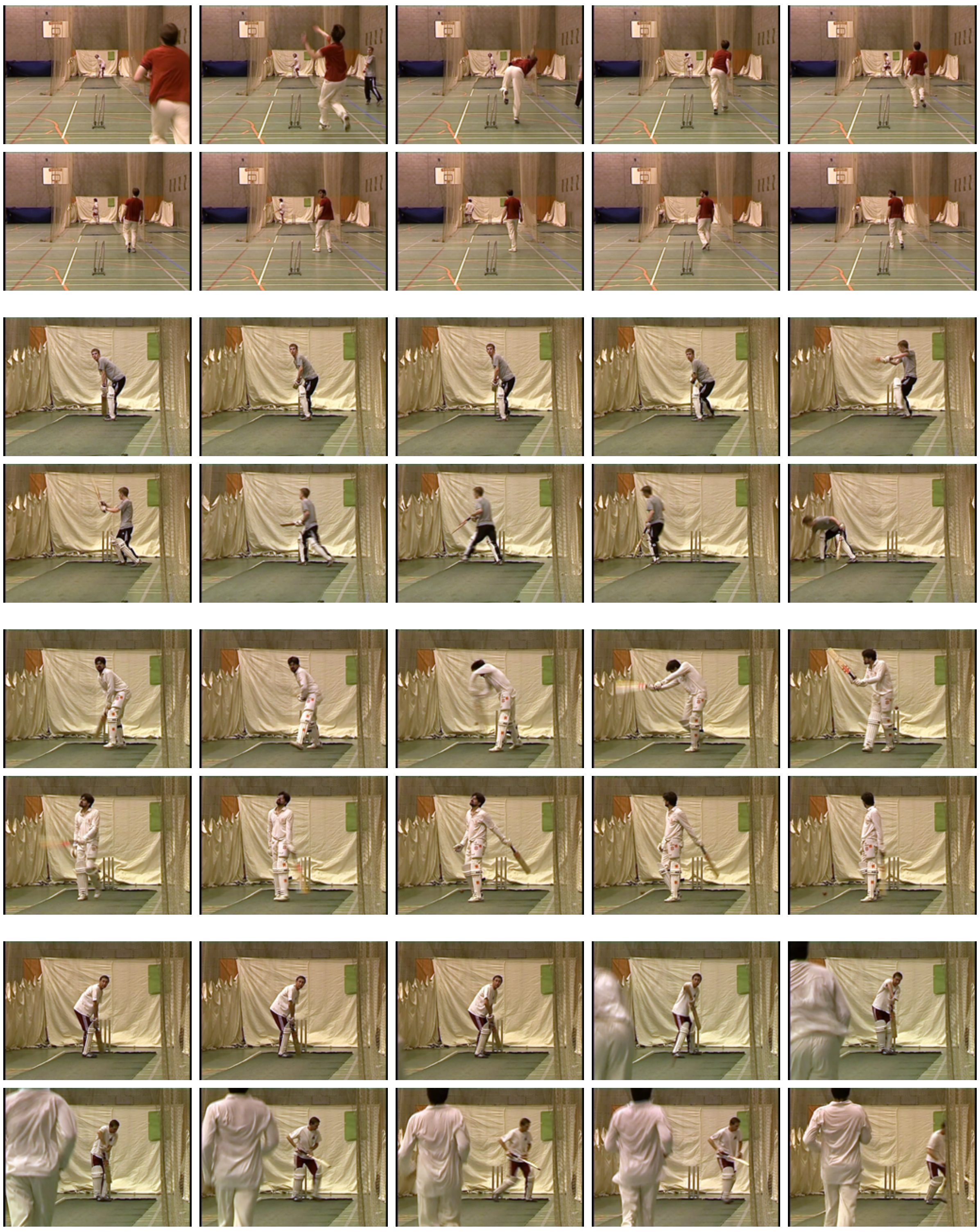


Figure 4.19: Examples of frames from single, correctly segmented, actions in the "Bowling", "Batting pt. 1", "Batting pt. 2" and "Batting pt. 3" sequences, shown in the four image sets, from top to bottom.

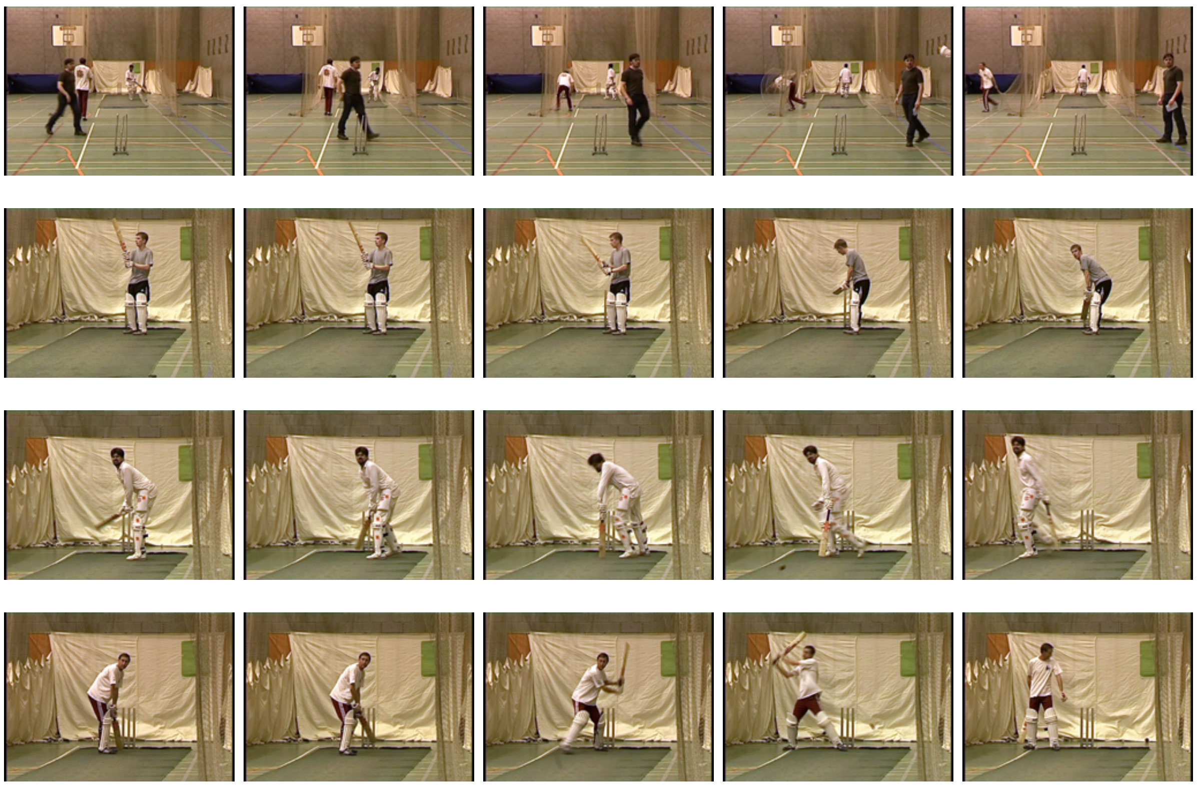


Figure 4.20: Example frames from falsely detected actions in the “Bowling” and “Batting pt. 1” sequences (first two rows), and missed actions in the “Batting pt. 2” and “Batting pt. 3” sequences (bottom two rows).

Chapter 5

A Review of Interactive Object Cut Out Techniques

The ability to automatically cut out an object in an image or video is one of the long standing challenges in image processing, resulting in a wealth of research being poured into the field. Many systems exist for automatically extracting objects in constrained environments, i.e. given particular motion patterns [182, 40, 38], flash lighting conditions [167], pre-learned objects [88] or other constraints [60]. However, in a video post production environment, no prior video conditions or scenarios can be assumed. In addition, the level of quality demanded by post production companies means that all segmentation work is performed at least partially by hand. Hence modern approaches aim to reduce the manual effort required by the digital artist.

The objective of a post-production cut out system is to enable compositing of the object. Digital compositing is "the digital manipulated combination of at least two source images to produce an integrated result" [31]. A common compositing scenario, in both still images and video, is that of placing an object from one scene into another. Another scenario is the selective application of a filter to only a part of an image. In both scenarios there needs to be some way of selecting or isolating the desired object or area in the image. This is termed "object cut out", and can be formally defined as the separation of an image into object and background components.

This chapter introduces the reader to the state of the art in semi-automatic object cut out, focusing particularly on how user knowledge is supplied and exploited. It begins by looking at the origins of video cut out, discussing the need for manual object cut out in post production, before introducing some modern methods for manual cut out in still images, and how they have influenced their video based counterparts. Modern post production video cut out systems can be broadly separated into two approaches; *spline based* "rotoscoping", and *pixel based* segmentation. The results of each produce a binary labelling specifying whether each pixel belongs to the object or the background, as shown in Figure 5.1. Following an initial extraction stage, most systems then apply a cut out refinement, known as "matting", to perform a non-binary labelling of pixels whose intensities contain a mixture of object and background.

Rotoscoping (hereafter abbreviated to *"roto"*) of video is the process of explicitly delineating the object by a series of curves in one frame, and manually manipulating the control points of the curves to fit the object in later key-frames. Between a pair of key-frames, the moving object is cut out through time by interpolating the positions of the curves. The general case of rotoscoping is an entirely manual process, no image data is used by the system to determine how or where the splines are placed. The advantage of this is that the digital artist maintains entire control over what is and is not considered object. This makes it the current favourite approach for object cut out in post production companies. However, consider the object cut out scenario where a clear delineation exists between the object and background, yet the object border is highly detailed. Although it is apparent how the object should be cut out, the artist must still manually place, push and touch-up many of the curve's control points. In difficult scenarios such as this, the amount of manual effort is high. In contrast, pixel based schemes attempt to model the object and background, and exploit some delineation between them to reduce the amount of user interaction required to extract the object.

Object segmentation refers to the per pixel labelling of object and background in an image. Automatic segmentation of objects has been a popular field of study for a long time. However, it is unlikely that any current fully automatic segmentation method would produce a segmentation of a quality high enough to meet the demands of the post production environment. Current approaches in object segmentation for post production realise this and attempt to keep the user in full control while automating the tedious parts. Semi-automatic approaches essentially let the user impart some information to the system, specifying what is object and background, for example with simple brush strokes. The system builds generalised object and background models using this information, and uses the models to accurately segment the current frame, or possibly subsequent video frames. The goal is to reduce the overall amount of explicit information the user needs to supply to segment the object in the video.

Matting Following an object cut out, a matting stage is typically performed to identify the non-binary pixel labelling, especially around the border of the object where a pixel may contain a mixture of object and background components. Some semi-opaque or translucent, pathological "objects" are composed entirely of ambiguous object and background pixels, such as smoke or glass. The level of object to background is typically denoted by alpha values, $\alpha \in [0, 1]$. As humans are not good at estimating how transparent something is, the matting is at least a semi-automatic process. In rotoscoping, to specify where the α values should be calculated, the artist can specify a band around the curves in which the pixel labelling is ambiguous. Similarly in the segmentation process a region of "unknown" pixel labellings can be specified around the object border.

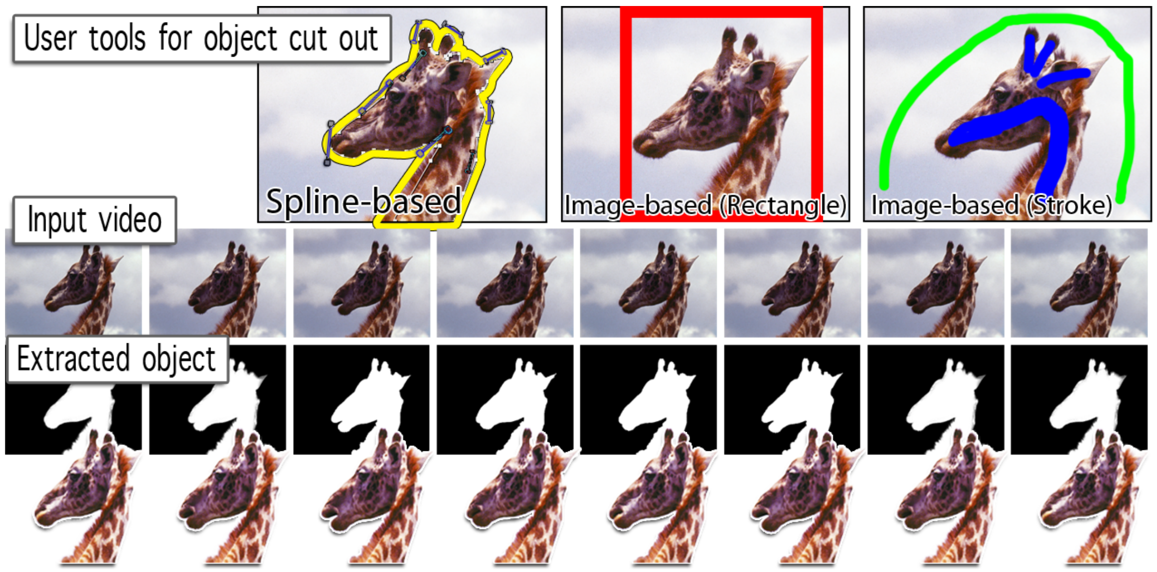


Figure 5.1: "Object Cut Out" is the process of labelling image regions as belonging to object, background, or somewhere in between. The majority of academic object cut out research is focused on semi-automatic image based segmentation schemes, while labour intensive spline based cut out techniques continue to be favoured in modern video post production companies.

The aim of this chapter is to introduce the reader to some of the state of the art in post production video object cut out schemes from both spline and pixel based paradigms. The disparity between the two is discussed in terms of cut out quality, the amount of time and effort required to extract a useful cut out, and how the user interacts with the system. In Section 5.2, popular methods of still image cut out systems are introduced. Section 5.3 builds on the 2D cut out systems, presenting some of the more interesting pixel and spline based video cut out systems. To complete the post production workflow of object cut out, the matting problem is introduced in Section 5.4, along with two popular solutions. Finally, in Section 5.5, the strengths and merits of the various systems presented throughout the chapter are discussed, where the motivation for using feature points for video cut out is developed. To put the field in context, a brief history of object cut out, matting and compositing in media post-production is now presented through a digest of two interesting books by Brinkmann [31] and Fielding [52].

5.1 Object Cut Out; A History

Today, advanced tools to cut objects out from still images have become standard, some of which are built into many of the default consumer image managers that accompany major operating systems (i.e. "Preview" or "iPhoto" in OSX, "The G.I.M.P." in Linux distributions). The first example of compositing still images is that of Oscar G. Rejlander's 32 photo amalgamation print "The Two Ways of Life", unveiled in 1857 [31]. 80 years later the 1933 film "King Kong" demonstrated the first use of compositing in cinema. For this film, stop-motion animations of

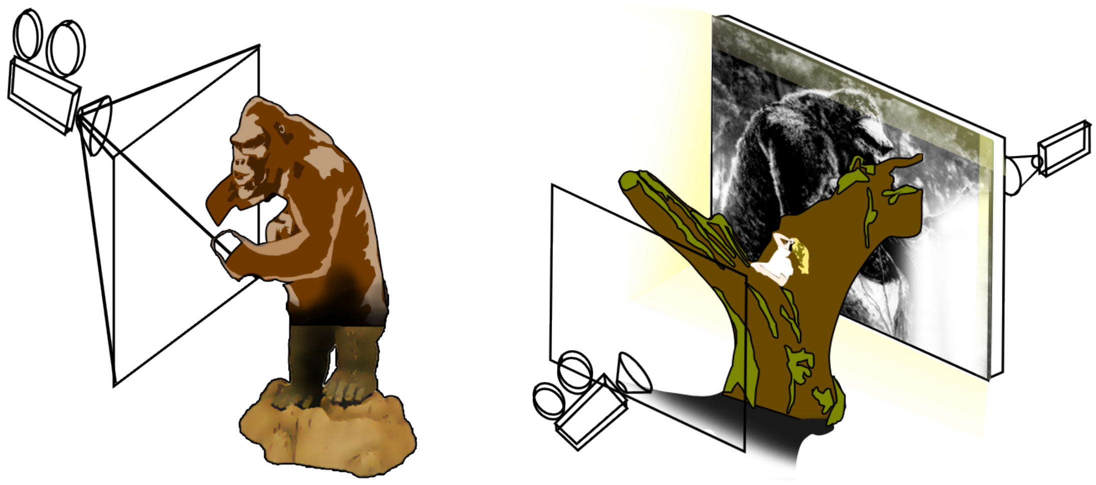


Figure 5.2: Early "in-camera" matting for cinema. To composite "King Kong" into real footage, the stop-motion animations were first filmed (left), then projected onto a large screen behind the real set (right). This is referred to as an "in-camera" process as no additional manipulation of the film is needed for the composition.

miniature King Kong models were filmed first in scaled down versions of the real sets, as shown in Figure 5.2. The miniature footage was then projected onto a large screen at the rear of the life-sized sets, giving the illusion of a monstrous ape existing in the same space as the real actors [31].

With compositing for film becoming a more common tool for movie producers, ways to automatically create realistic mattes began to be investigated. Early matting techniques involved simultaneously recording a subject with different types of film that were responsive to non-visible light, for example one camera would capture an infrared or ultraviolet image, while another would capture the visible light [52]. By lighting the subject with regular lights, and a background screen with strong non-visible light, the film responsive to the non-visible light would clearly show which parts of the image belonged to the foreground and background. This automatic approach allowed difficult cinematic elements such as fog and smoke to now be incorporated (which were impossible to draw by hand), giving a much greater feeling of realism to the composition.

During the early 1960's, Disney sought to make a more convenient matting system. Their idea was to use monochromatic sodium vapour light instead of using non-visible light [72]. The wavelength of sodium light is so narrow that most standard colour film stock at the time was not sensitive to it. This allowed the set to be lit with high power sodium lamps, giving a clearer response (and clearer matte) on the sodium film stock, without any effect on the visible light stock, an example is shown in Figure 5.3. Disney also addressed the problem of requiring two separate cameras to acquire a matte. They began experimenting with combining two recording reels into the same camera and using a prism beam splitter to project the same light onto a standard colour film stock and a sodium light sensitive film simultaneously [52]. As the sodium

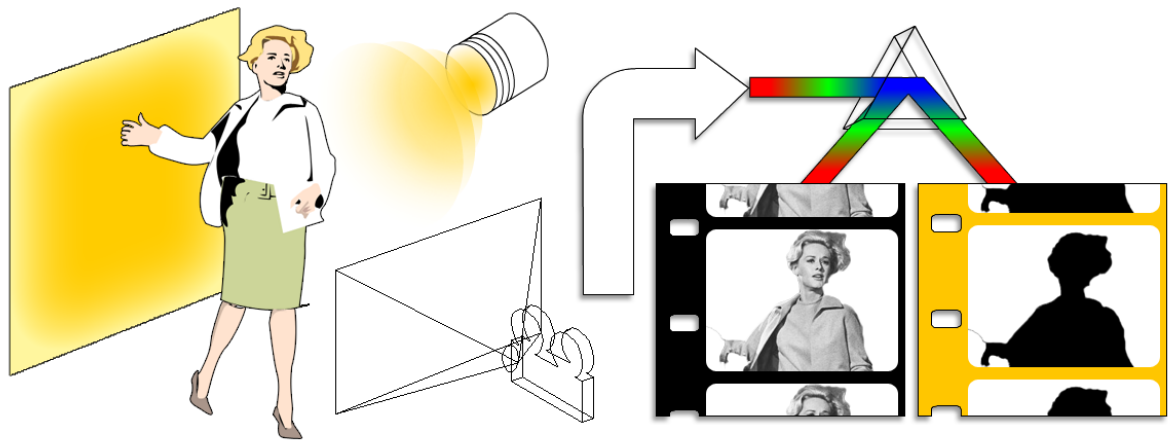


Figure 5.3: Example of the "sodium vapour process". The actor is filmed in front of a background screen of strong monochromatic, sodium lamps. A prism splits the incoming light onto two film reels simultaneously, one reactive to regular light, and the other to sodium light wavelengths, resulting in the shot of the actor, and the associated sharp matte.

sensitive stock was of the same physical size, the matte did not need any optical pre-processing before it could be used for compositing, which was a huge advantage.

In the modern digital age, the analogous technique of *chroma-keying* (sometimes known as "difference matting") automatically creates a matte of a subject filmed against a green or blue screen [76]. With prior knowledge of the specific blue or green hue, the distance in colour-space between the colour intensities of a pixel and the background colour is used to estimate the amount of background present at that pixel site. An example is shown in Figure 5.4. Under proper lighting conditions chroma-keying produces very sharp mattes, and is a convenient, inexpensive method of isolating an object from the background; most "pro-sumer" (professional consumer) video editing packages, such as Final Cut or Adobe Premiere, include a chroma-keying tool by default.

However, problems such as lighting differences, and bleeding of the blue / green colour onto the subject, make it difficult to escape the artificial look of the resulting composition. As such there has been a growing trend to instead extract objects from "naturally" filmed scenes. It may not be possible to film on a blue screen for a number of reasons, for example the scene may be too complicated, the budget may be limited, more natural light is desired for the shot, or that the shot has already been filmed without a blue screen. An example of natural scene object cut out is shown in Figure 5.5. However, due to the diversity of natural scenes, no simple, fully automatic system exists and so the process relies on user interaction. Digital compositing in still images is a mature field, many of the recent advances in video object extraction and manipulation are direct extensions of still image techniques. The following section introduces the reader to some of the state of the art in still image object cut out methods, as they become relevant to video based techniques later on.

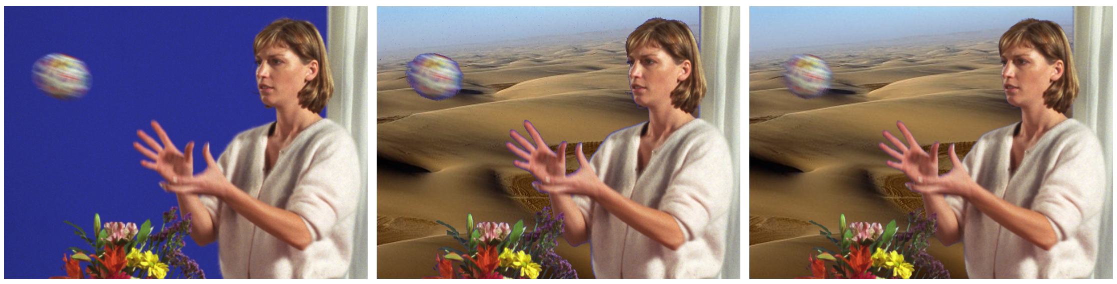


Figure 5.4: Example of a difficult chroma-keying scenario. Left, the motion blur of the ball, combined with poor lighting conditions makes this a difficult task for chroma-keying. Centre, an initial automatic cut out, notice the labelling ambiguity induced by the motion blur. Right, user correction is applied to the difficult regions and the situation improves, some colour "bleeding" remains on the ball boundary.

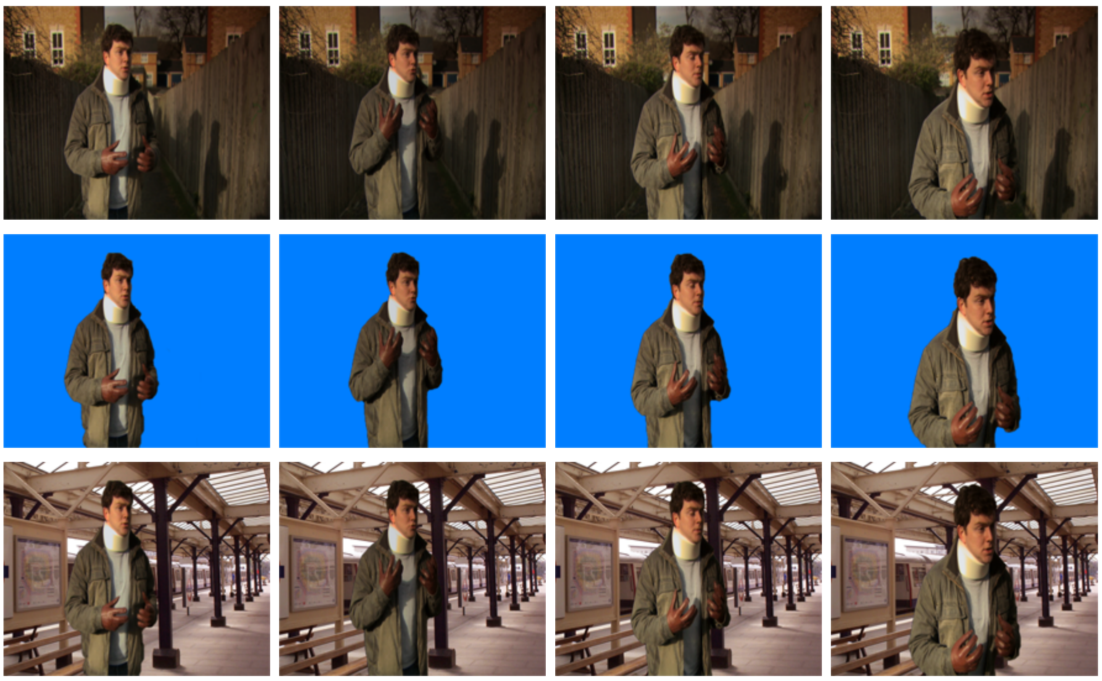


Figure 5.5: The objective of semi-automatic "natural" cut out; to extract an object from a real scene (2nd row), either to allow (in theory) more realistic compositing (3rd row), or for selectively applying filters to the object.

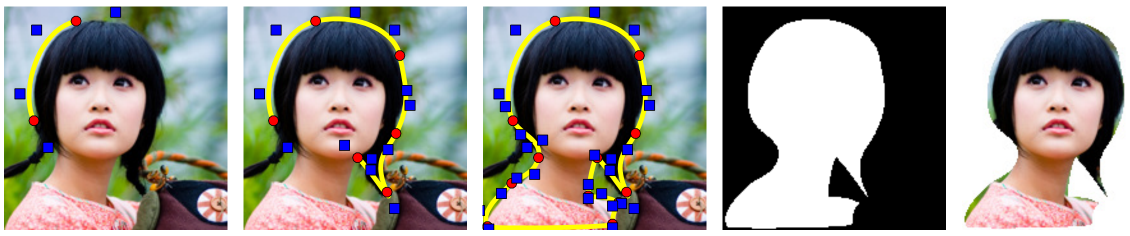


Figure 5.6: Example of using Bezier curves (sometimes known as "paths") to cut out an object. First image, a single spline is created between two (red circle) points, using the locations of the two control points (blue squares) to influence the shape of the curve. Second and third images, more splines are placed to better delineate the subject. Fourth and last image, the object cut out is given by the shape of the closed curve. Notice that although the cut out appears smooth, more manipulation of the curve's points is required, as parts of the background were included, and parts of the subject were omitted.

5.2 Still Image Segmentation

There has always been a demand for manipulation of objects within natural scenes, such as for touching up colour, or applying filters selectively to parts of an image as part of a typical compositing workflow. This has resulted in a number of tools developed to reduce the manual effort required to selectively extract parts of a still image. Two distinct techniques stand out, in particular the use of "paths", and selection by colour. "Paths" is the still image equivalent of rotoscoping, i.e. specifying an object boundary by manually drawing connected curves using control points [1], an example is shown in Figure 5.6. Like rotoscopes, paths give the user direct control over what is accepted as object or not, without considering actual pixel values.

Colour selection is another popular object extraction method, in which the user first specifies a range of colours for the object, perhaps by sampling directly from the image itself. Like chroma-keying, the algorithm determines what pixels are considered object by how close the colour of each pixel is to the range of sampled colours. A comparative example between using paths and colour to select the object is shown in Figure 5.7. This simple colour approach works well when the colour distributions between object and background are sufficiently separated, but fails as soon as there is any overlap in distributions. However, colour can be a very useful feature, and is used to great effect in modern, state of the art object cut out systems. The following two still image cut out schemes have been selected for discussion as they represent milestones in segmentation development, and are relevant to later work in this thesis. Both systems employ non-binary matting stages following the initial "hard", binary segmentation. For the moment, the emphasis is placed on how the objects are found and (binary) segmented in the image.

All successful segmentation algorithms can be unified with a Bayesian framework. The goal is always to separate an image into foreground, $\alpha(\mathbf{x}) = 1$, and background, $\alpha(\mathbf{x}) = 0$, components. The label field $\alpha(\mathbf{x})$ then represents the segmentation. In the process of "matting", α is considered to the a continuous scale bounded between 0 and 1. Hence, α-mattes ("alpha

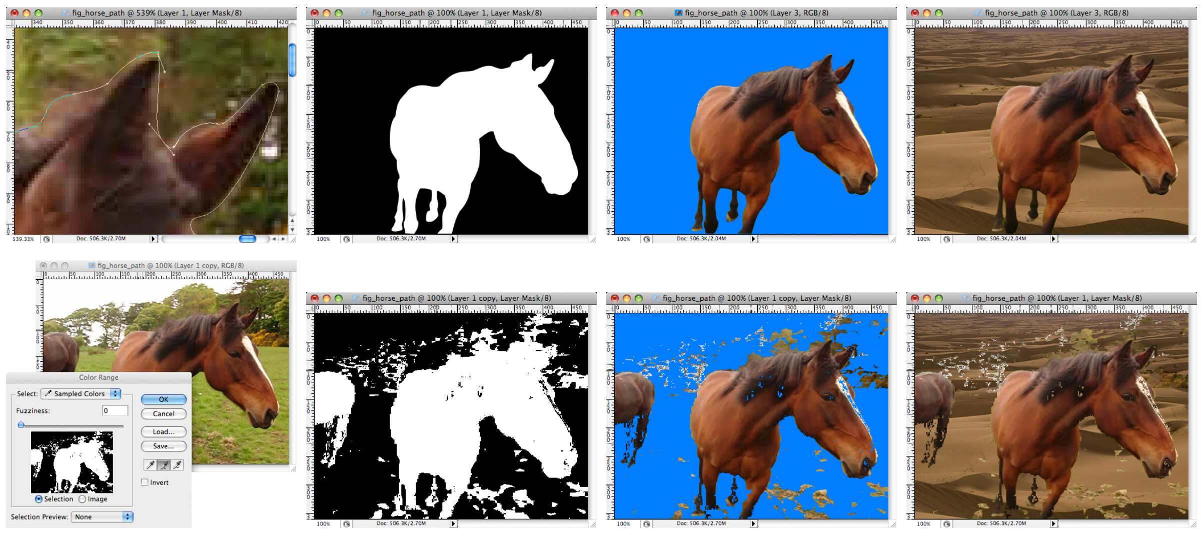


Figure 5.7: Two popular techniques for still image cut out; paths (top row) and colour selection (bottom row), using Adobe Photoshop. The first column shows the use of Bezier curves to delineate the horse's ears from the background (top). A colour range that represents the horse is selected (bottom). Notice the segmentation produced by the paths is of a higher quality matte than using colour selection (second column, top and bottom). The errors in the colour matte are due to the horse exhibiting similar colours as the background. Notice however, that in the composite image using the path cut out (last column) some green background pixels were included, for example, under the jaw of the horse. Both mattes need some correction. The times taken to produce the mattes were approx. 5 minutes using paths and 10 seconds for colour. In practical terms it is not clear which method is "better", as the time saved using colour selection could be used to correct the resultant matte.

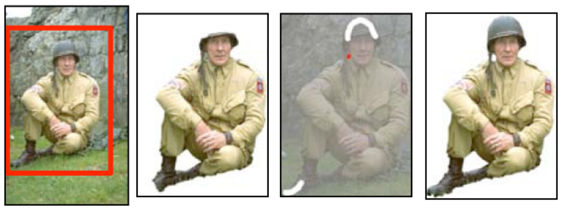
mattes") tend to model ambiguity of object boundaries as well. Continuing in a probabilistic fashion, the problem can be posed as estimating the best α that maximises $P(\alpha|I,\theta)$, where I is the observed image data, and θ are some system parameters. Using a Bayesian approach this becomes:

$$P(\alpha|I,\theta) \propto P(I|\alpha,\theta)P(\alpha|\mathbf{C})P(\theta) \tag{5.1}$$

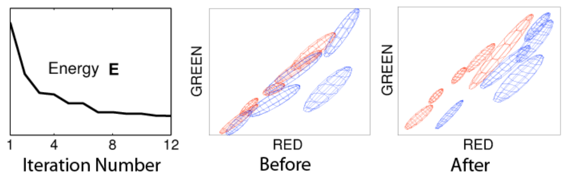
where $P(I|\alpha,\theta)$ is the data likelihood, and the terms $P(\alpha|\mathbf{C})$ and $P(\theta)$ encode prior knowledge of the system and the label field α. In the presented segmentation applications, the prior term $P(\alpha|\mathbf{C})$ is used to encourage the label field α to be smooth within the neighbourhoods defined by $\mathbf{C}$ (unless stated otherwise). All techniques can be understood through (i) assignment of likelihood and prior and (ii) the adopted optimisation strategy for α. Instead of dealing in probabilities, it is common to use the negative logarithms of the probabilities in Equation 5.1,

$$E(\alpha,\theta,I) = U(\alpha,\theta,I) + \lambda V(\alpha,I,\mathbf{C}) \ . \tag{5.2}$$

Maximising the original posterior $P(\alpha|I,\theta)$ now becomes minimising the energy of $E(\alpha,\theta,I)$, where U and V are the data and spatial energies, and λ is used to weight the contribution of each.



(a) User Interaction



(b) Refinement Process

Figure 5.8: Example of using the Grab-Cut system. Left, the user marks an area in the image containing the object (1st image), and the system proceeds to extract the object from the background (2nd image). To correct any mistakes, the user places "scribbles" explicitly defining foreground and background (3rd image), and the segmentation is run again (4th image). Right, the label assignment energy (cost) decreases over multiple iterations (1st image). Notice that the colour distributions are better separated after the refinement process (2nd and 3rd images).

To reduce repetition and improve the clarity, the interesting and relevant parts of the following still image and video segmentation frameworks are discussed in terms of the key elements of the Bayesian framework; data likelihood $U(\alpha, \theta, I)$, spatial prior $V(\alpha, I, \mathbf{C})$, and optimisation strategy. To begin, an impressive framework is presented for selecting objects in still images by Rother et al. [148] known as "Grab-Cut", based on previous work by Boykov & Jolly [25].

5.2.1 Grab-Cut

The "Grab-Cut" system by Rother et al. [148] allows the user to draw a rectangle in the image enclosing the entire object, and probably also containing background regions, as shown in Figure 5.8a (far left image). The objective then is to figure out which parts of the image inside the rectangle definitely belong to the object and background. This is accomplished by iterative segmentation and re-modelling of colour distributions.

Likelihood Following the initial user input, the image is partitioned into the foreground / background mixture region within the rectangle, T_U, and the rest of the image, T_B. The data likelihood of the label field is given by the colour distributions in the T_U and T_B regions. These colour distributions are modelled as Gaussian Mixture Models (GMM), iteratively fit using Expectation-Maximisation (EM). Rother et al. use two GMMs, one each to model the colour in the T_U and T_B regions, with $K = 5$ components in each model. Each pixel in the image is assigned to a single component, k_n from either the object or background models (depending on the current α labelling), the assignment is denoted by $\mathbf{k} = \{k_1, \dots, k_N\}$ and $k_n \in \{1, \dots, K\}$. The model variables are given by:

$$\theta = \Big\{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k), \alpha = \{0, 1\},\ k = \{1 \dots K\}\Big\} \tag{5.3}$$

where $\pi(\alpha, k)$ is the contribution (weighting) of the component k to the GMM, $\mu(\alpha, k)$ is the vector of mean colour channel intensities for component k, and $\sigma(\alpha, k)$ is the covariance of the colour channel intensities of component k to the object ($\alpha = 1$) or background ($\alpha = 0$) GMM model. Using the model parameters θ, the probability of observing the image data $I(\mathbf{x}_n)$ (at site $\mathbf{x}_n$) given α_n, and k_n is given by the multivariate Gaussian function,

$$P(I(\mathbf{x}_n)|\alpha_n, \theta_n) = \pi(\alpha_n, k_n)\mathcal{N}\Big(\mu(\alpha_n, k_n), \Sigma(\alpha_n, k_n)\Big) \tag{5.4}$$

The joint likelihood over all pixels in the image is a product of each likelihood term. The energy term U of the data likelihood from Equation 5.2 is therefore found by summing over the negative log likelihood of $P(I(\mathbf{x}_n)|\alpha_n, \theta_n)$:

$$\begin{aligned} U(\alpha, \theta, I) = &- \sum_n \log \pi(\alpha_n, k_n) + \frac{1}{2}\sum_n \log\left(|\Sigma(\alpha_n, k_n)|\right) \\ &+ \frac{1}{2}\sum_n \left([I(\mathbf{x}_n) - \mu(\alpha_n, k_n)]^T \Sigma(\alpha_n, k_n)[I(\mathbf{x}_n) - \mu(\alpha_n, k_n)]\right) \end{aligned}$$

Prior The spatial prior term V encourages smoothness in the segmentation labelling by penalising neighbouring pixels with different labellings. Grab-Cut uses a Markov Random Field (MRF) to model the spatial connectivity between the pixels, each pixel is connected to each of its 8 nearest neighbours. The set of all pixel cliques in the image is denoted $\mathbf{C}$. The spatial energy term V is then defined as

$$\begin{aligned} V(\alpha, I, \mathbf{C}) = &\quad \gamma \sum_{(m,n)\in\mathbf{C}} [\alpha_n \neq \alpha_m] \exp -\beta \|I(\mathbf{x}_m) - I(\mathbf{x}_n)\|^2 \\ \beta = &\quad \left(2 \cdot E[(I(\mathbf{x}_m) - I(\mathbf{x}_n))^2]\right)^{-1}, \ \forall (m, n) \in \mathbf{C} \end{aligned}$$

where $E[]$ is the expectation operator. The idea is to encourage $\alpha_n = \alpha_m$ when the underlying image data is smooth, i.e. when the image gradient between $I(\mathbf{x}_n)$ and $I(\mathbf{x}_m)$ is low. β simply controls how much gradient is allowed to influence the smoothness.

Optimisation Strategy With the terms U and V defined, the energy Equation 5.2 can be posed as a min-cut / max-flow graph problem. The Graph-Cut algorithm [87, 29, 28] is then used to find the labelling field $\hat{\alpha}$ with the lowest energy, i.e.

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \ E(\alpha, \theta, I) \ . \tag{5.5}$$

However, one of the novel aspects of Grab-Cut is how the user supplied information is exploited to maximum effect by performing multiple energy minimisation iterations of Equation 5.5. First α is optimised with respect to the model parameters θ to give a putative label field in T_U, partitioning regions in T_U as belonging to the object, T_F ($\alpha = 1$) or belonging to the background, T_B ($\alpha = 0$). The modelled colour distributions are updated to reflect this estimated label field using the T_F and T_B regions, i.e. θ is optimised with respect to α. Grab-Cut alternates between these two optimisations until convergence (there are no more changes to the labelling fields T_F or T_B), with the resulting segmented object region given by T_F.

Using the estimated labels as input for colour model refinement causes the colour distributions for object and background to separate, resulting in a better segmentation. If the calculated regions T_U or T_B contain errors, the user is allowed to manually "stroke" the image, constraining the α values at pixel sites under the stroke. An example of user interaction and the effects of the iterative refinement process are shown in Figure 5.8. This automatic separation of colour distributions is one of the strengths of Grab-Cut. There is often debate over the choice of colour space to use when using colour as the primary feature. Rother et al. acknowledge this, and say for their purposes, the standard RGB space is sufficient. However, if desired the scheme could easily be adapted to use a less correlated colour space, such as Lab^* or $YCbCr$, using the chrominance components to estimate the likelihood (i.e. $U(\alpha, \theta, \{Cb, Cr\})$), and the luminance channel to estimate the per-pixel neighbourhood prior (i.e. $V(\alpha, Y, \mathbf{C})$).

Another strength of Grab-Cut is the implicit use of spatial information. For example, when the user marks a rectangle in the image, the pixel sites outside the rectangle (or under a "background" scribble in T_U) are given infinite costs to be labelled as object. During energy minimisation, the smoothness constraint of the MRF encourages neighbouring pixels to adopt the same label, particularly if the local image gradient is low. This has the effect of implicitly introducing proximity to the rectangle boundary as a likelihood, allowing parts of the background to have the same colour as the object and still be successfully segmented, providing the background region with the same colour is close to the rectangle. This proximity is not encoded directly in the MRF, but the notion of explicit proximity likelihood is very powerful, and is extended by the Distance-Cut object segmentation scheme of Bai & Sapiro [10, 9].

5.2.2 Distance-Cut

As with Grab-Cut, the likelihood $P(I|\alpha)$ of each pixel in the image belonging to either object or background is estimated from user delineated areas, supplied as paint strokes. The novelty of the Distance-Cut system is that proximity to the user strokes is used to augment the data likelihood given by the colour distributions.

Likelihood The pixel regions given by the foreground F and background B user supplied strokes are denoted Ω_F and Ω_B. Fast KDE techniques [186] are used to model the underlying colour distributions given the empirical histogram measured from Ω_F and Ω_B. Bai & Sapiro use a likelihood ratio to generate a new object and background likelihood over all pixel sites $\mathbf{x}$,

$$\begin{aligned} P_F(\mathbf{x}) &= \frac{P(I(\mathbf{x})|\alpha = F)}{P(I(\mathbf{x})|\alpha = B) + P(I(\mathbf{x})|\alpha = F)} \\ P_B(\mathbf{x}) &= 1 - P_F(\mathbf{x}) \ . \end{aligned}$$

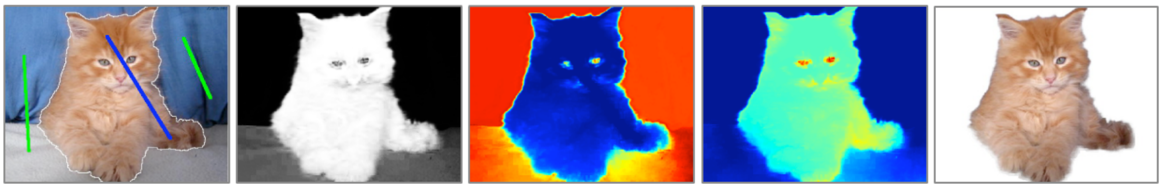


Figure 5.9: Example of using the Distance-Cut system. The user begins by drawing object (blue) and background (green) scribbles (1st image). The system calculates the label likelihood for each pixel from the object and background colour distributions (object colour likelihood, $P_F(\mathbf{x})$, shown in 2nd image). The likelihoods are then used as weights in the object and background geodesic distances (3rd and 4th images, blue is low distance, red is high). The labelling at each site is given by the class with the minimum geodesic distance at that site (last image). Note that the colour likelihood in the second image is probably good enough to allow segmentation without using geodesic distances. (Images reproduced from Bai & Sapiro [10].)

Prior In this algorithm, an explicit spatial prior is not used. Instead the likelihood distributions, P_F and P_B, are mapped into a different space based on generalised geodesic distances. The geodesic distance from one site s to another t in an image I is given by

$$\begin{aligned} d(s,t) &= \min_{C_{s,t}} \int_0^1 W dp \\ W &= |\nabla I \cdot C'_{s,t}(p)| \end{aligned} \tag{5.6}$$

where C is the path from s to t, parameterised by $p \in [0,1]$, $\bigtriangledown$ is the 2D gradient operator, C' is the first derivative of C. Intuitively, $d(s,t)$ calculates the distance of the shortest path between s and t, where the distance of the path is a function of image gradients along the path. Geodesic distances have been proposed for video segmentation in the past [26, 157]. However, Bai & Sapiro calculate two geodesic distances in the object and background data likelihoods, P_F and P_B, from the foreground and background strokes respectively,

$$d_\alpha(\mathbf{x}) = \min_{s \in \Omega_\alpha} d(s,\mathbf{x}), \quad \alpha \in \{F, B\} \tag{5.7}$$

where $P_F(\mathbf{x})$ and $P_B(\mathbf{x})$ take the place of the "image" Y in Equation 5.6. The data energy is simply given by the geodesic distances:

$$U(\alpha,\theta,I,\mathbf{x}) = \begin{cases} d_{\alpha=1}(\mathbf{x}) & \text{if} \quad \alpha = F \\ d_{\alpha=0}(\mathbf{x}) & \text{if} \quad \alpha = B \end{cases}$$

Optimisation Strategy The object labelling at each pixel site is given simply by the minimum of the object or background energy at that site,

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \; U(\alpha,\theta,I) \; .$$

An example showing the geodesic distance is shown in Figure 5.9. The advantages of the Distance-Cut system are two-fold. Firstly, spatial information is intuitively incorporated into the system through geodesic distances, i.e. the object and background are allowed to have very similar colours, providing those coloured regions are spatially close to a user scribble. Secondly, the system is computationally efficient: The number of pixel sites sampled by the system is much lower than Grab-Cut (i.e. the area of the user scribble marking the object is significantly lower than a box bounding the object). Combined with an optimised algorithm for calculating geodesic distances in near linear time, the Distance-Cut system is very fast, even on large ($> 2M$ pixel) images. This speed allows label corrections to be applied rapidly and interactively.

The Grab-Cut and Distance-Cut are two impressive systems for extracting objects in still images in terms of framework design and quality of results. The frameworks used in both systems have been extended for use in video. The following section of this chapter presents a discussion of the state of the art in extracting objects from natural video, comparing spline based techniques such as rotoscoping to the pixel based video extensions of Grab-Cut and Distance-Cut, and others.

5.3 Rotoscoping vs. Segmentation in Video

Spline- and pixel based are diametrically opposed approaches to object cut out. Spline based methods favour meticulous manual labour over computational assistance in extracting objects from their backgrounds. Very little is automated in the spline based cut out workflow. Although the manual effort involved in obtaining even an average quality matte in this way is extraordinarily high for an inexperienced user, skilled operators can achieve high quality cut outs reasonably quickly [31]. Pixel based techniques on the other hand remove the user from the process once initial user information is supplied, allowing the computer to do the heavy lifting. Both systems operate on the idea of “key-frames”; where information is supplied in one frame, and propagated somehow to other frames. For example, in the spline based approach, Bezier curves may be drawn in a pair of temporally disparate frames. The application can then interpolate the locations of the curve’s control points between the key-frames, automatically giving a rough placement of the curves throughout the intermediate frames, reducing the amount of work required by the operator. In the case of the pixel based schemes, key-frames refer to the frames where the user inputs information to initialise or correct a segmentation, for example, scribbles denoting object or background regions or colours. The manual effort of extracting an object can be lowered in two ways; by reducing the number of key-frames needed, or reducing the amount of effort per key-frame.

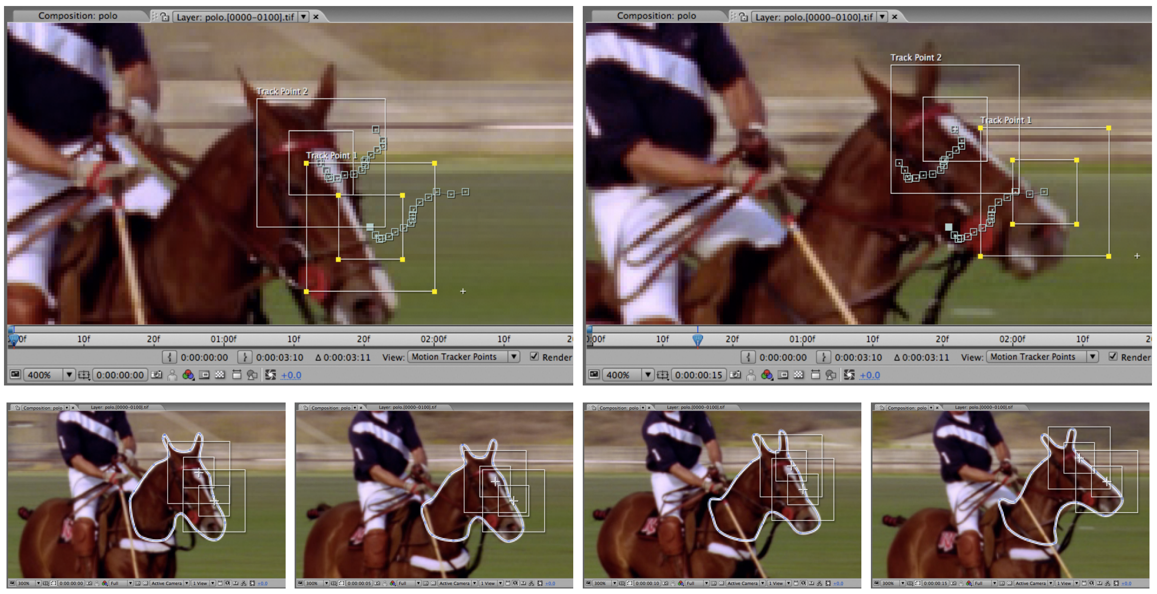


Figure 5.10: Example of point trackers in Adobe's After Effects program. Top row, two block based point trackers (white squares) are initialised on the horse's head (left), and allowed to track the head motion over 15 frames (right). The locations of the tracked points can be seen by the trails of small green squares. Bottom row, the points tracked are used as locations for two control points belonging to the Bezier curve outlining the horse's head (white), reducing the amount of manual effort required to delineate the horse in the 15 frames. (The remaining control points were moved by hand in each frame.)

5.3.1 Spline based Cut Out

In spline based systems, the quality of the cut out is a product of the skill of the user, the complexity of the sequence, and the time allotted, making it very simple for a post production company to estimate either the time cost for a matte of a desired quality, or the quality of a matte given a fixed amount of time. This also means if a tool can reduce the time required to create a video matte of a certain quality, the post production company saves money. Research and development into improving object cut out and matting tools is therefore important. The strategies, focus and state of the art of spline based cut out systems are now presented.

On the whole, improvements in spline based methods have mostly focused on the user interface, streamlining curve manipulation, frequently used commands or allowing the user to tweak the way curve control points are interpolated between key-frames to minimise the overall number of key-frames required [2, 75]. This speeding up of the artist's *micro*-workflow makes sense, it is the key-frames that cost the company time, if the number (or the time taken to make one) can be reduced, the company saves time. This makes it cost effective to automate the tedious parts of rotoscoping.

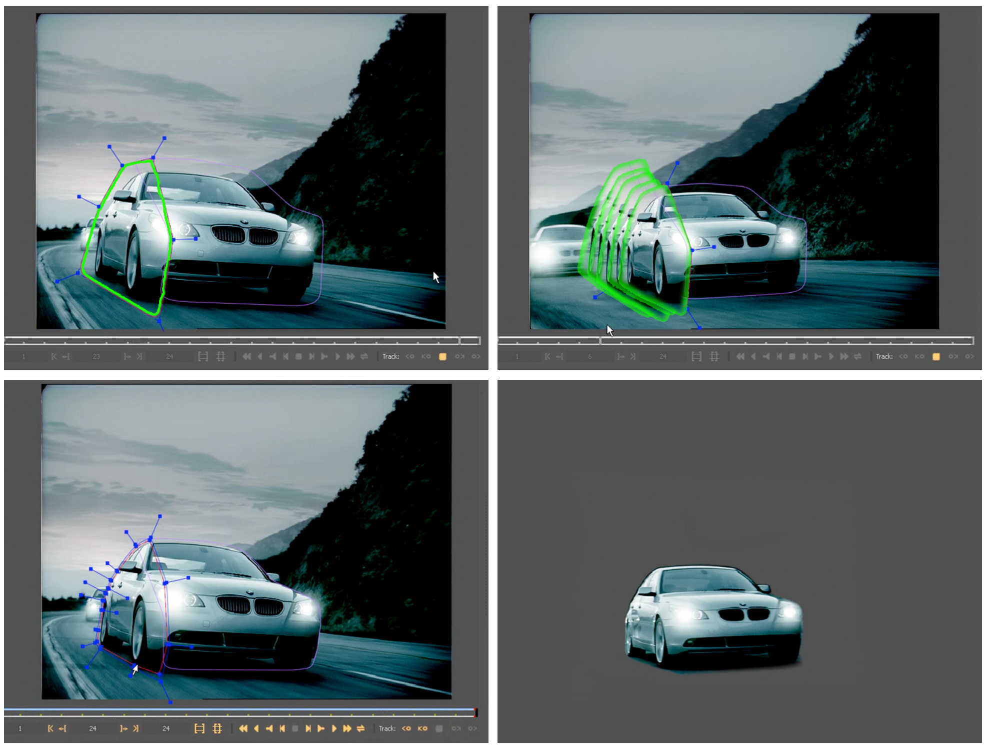


Figure 5.11: Example of planar tracking in Imagineer Systems' "Motor". In the first image, a coarse region (corresponding to the side of the car) in frame 23 is selected to be tracked (green). Second image, the user assumes a perspective motion model, and allows the system to track the region backwards through the sequence (to frame 6), warped in-between regions are shown in green. Third image, a more accurate set of roto curves are placed in the same region at frame 23. The motion parameters from the region track are applied to the accurate curve set. A separate track is calculated (and used to drive another refined curve set) for the front of the car (as it is assumed the side and front of the car are allowed to deform separately). Last image, the accurate matte in frame 6 is created entirely using the backwards propagated roto information. (Images courtesy of Imagineer Systems.)

5.3.1.1 Roto Point Tracking

Of the techniques involved in expediting object cut out, simple point tracking has proved to be one of the most popular. This involves the user assigning a block based point tracker to one or more curve control points, and allowing the tracker to find the corresponding point in subsequent images, an example is shown in Figure 5.10. The idea is to estimate the geometric transform between two regions in consecutive frames, and apply that transform to local curve control points. In the After Effects example shown in Figure 5.10, the user can track the location, scale and rotation or perspective transform of a particular point. This technique is useful in speeding up tedious rotoscoping tasks, however the tracker usually only works correctly in uncomplicated scenes. In many cases the roto artist will still need to refine these tracked points by hand. However, there are some rotoscoping situations in which the properties of the object can be

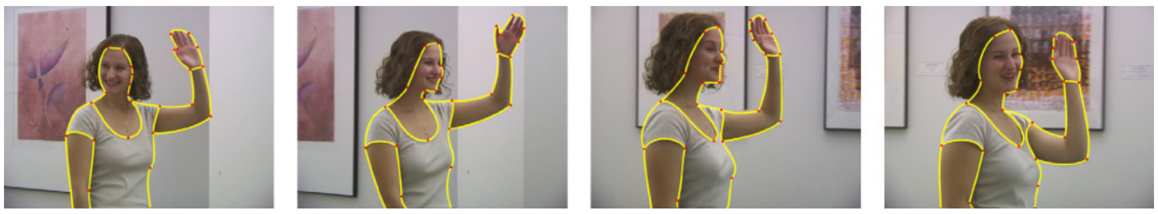


Figure 5.12: Example of roto curves accurately tracking the subject over a three second sequence using a smart energy minimisation scheme. (Images reproduced from Agarwala et al. [3].)

exploited. Consider a scene where objects ungergo rigid deformation corresponding to a planar motion model, i.e. affine, perspective, etc. These conditions present problems for roto artists, where it appears obvious that points belonging to an object region are moving together, yet key-frames need to be manually created and curve control points manipulated to match the motion warp. It may be simple to manipulate all the points at once if the object is simply rotating or changing scale or location, however for dramatic perspective warps many of the points need to be moved by hand, paying particular attention to ensure the point motions are visibly smooth through time as well.

5.3.1.2 Roto Planar Tracking

Recently, Imagineer Systems' "Motor" program introduced planar tracking of Bezier shape regions to help automate rotoscoping objects in these obvious but difficult scenarios. In the semi-automatic planar tracking scenario, a large, coarse object section of the moving region is specified by the user. This is typically some region where the object is expected to *rigidly* deform, i.e. the majority of points in the region undergo the same transformation. A motion model (affine or perspective) is then fit between the user selected region in one frame, and the corresponding region in the next frame. The model may be fit using pixel or feature correspondences between the image pair, or a coarse-to-fine mixture of the two. The resulting motion parameters are then used to guide a more accurate spline shape, which typically overlaps with the region used for tracking. The idea is that the detailed object curve should only need to be drawn once, and use the tracked motion model to push the accurate splines throughout the sequence. An example is shown in Figure 5.11. Planar tracking is a relatively low-tech image processing technique by modern tracking standards, however its success lies in its reliability, speed and ease of use; if the track appears to be drifting, it is simple to correct and restart the track. Assuming rigid object deformation, this sort of region tracking is more robust to changing image conditions than point tracking, and can greatly reduce the number of required key-frames.

5.3.1.3 Roto Key-frame Tracking

Realising that creating key-frames are the expensive part of rotoscoping, Agarwala et al. propose a system that performs more intelligent interpolation between key-frames by tracking video at the curve control points and exploiting the connected nature of points [3]. Video tracking has been used to help automate rotoscoping in the past [34, 73]. The novelty of Agarwala et al. is the use of a smart energy minimisation framework for propagating the locations and shapes of the curves to the frames between a pair of key-frames using the image data. The energy model is broken into "shape" and "image" terms, the latter being used for the data likelihood, and the former encouraging spatial consistency,

$$E = \underbrace{w_L E_L + w_C E_C + w_V E_V}_{\text{Shape terms}} + \underbrace{w_I E_I + w_G E_G}_{\text{Image terms}}$$

where w_L,w_C,w_V,w_I,w_G weight the influence of each of the terms.

Consider a curve $\mathbf{c}_t$ at time t, where the position along the curve of N_s segments (from start to end) is given by $i \in \{1, \ldots, N_s\}$, with the parameters of the curve the curve given by s. The location (in the image plane) of each point along the curve is therefore given by $\mathbf{c}(s_i)$. Consider that the curve exists in two temporally disparate key-frames at times a and b, and that the curve undergoes some transform between the key-frames. The goal is to determine the curve parameters s for all the intermediate frames using the Equation 5.3.1.3. The shape terms penalise changes in the curve length (E_L), curvature (E_C), and high between-frame velocity (E_V), and are based strictly on the spatial information of the curves,

$$\begin{aligned}
E_L &= \sum_{i,t} \left(\|\mathbf{c}_t(s_{i+1}) - \mathbf{c}_t(s_i)\|^2 - \|\mathbf{c}_{t+1}(s_{i+1}) - \mathbf{c}_{t+1}(s_i)\|^2 \right)^2 \\
E_C &= \sum_{i,t} \| \left(\mathbf{c}_t(s_i) - 2\mathbf{c}_t(s_{i+1}) + \mathbf{c}_t(s_{i+2})\right) - \\
&\quad \left(\mathbf{c}_{t+1}(s_i) - 2\mathbf{c}_{t+1}(s_{i+1}) + \mathbf{c}_{t+1}(s_{i+2})\right) \|^2 \\
E_V &= \sum_{i,t} \|\mathbf{c}_t(s_i) - \mathbf{c}_{t+1}(s_i)\|^2 \ .
\end{aligned}$$

The image terms specify how the curves are "tracked" by penalising curve configurations with differences in image areas around the curve,

$$E_I = \sum_{i,k,t} \|I_t\left(\mathbf{c}_t(s_i) + k\hat{\mathbf{n}}_t(s_i)\right) - I_{t+1}\left(\mathbf{c}_{t+1}(s_i) + k\hat{\mathbf{n}}_{t+1}(s_i)\right)\|^2$$

where I_t is the image at frame t, $\hat{\mathbf{n}}_t(s_i)$ is the unit normal tangent vector at the curve $\frac{d\mathbf{c}_t(s_t)}{ds_i}$ rotated 90°, i.e. the perpendicular direction from the curve $\mathbf{c}_t$ at s_i. The variable k is some user specified search radius, set to $k \in \{-5, \ldots, 5\}$ for double-sided tracking, 5 pixels either side of the curve for example, or $k \in \{0, \ldots, 5\}$ for tracking of a single side only. A tracking "lock" is then encouraged by requiring that the gradients along the curve at each frame during the track should be similar to the gradients along the curve at both key-frames,

$$E_G = \sum_{i,t} \left(\frac{G'(\mathbf{c}_t(s_i))}{M_i} \right)^2$$

where $G(\mathbf{p})$ is the image gradient at the point $\mathbf{p}$, $G'(\mathbf{p}) = G(\mathbf{p}) - K$, K is the maximum gradient along the curve, and $M_i = \min(G'(\mathbf{c}_{t=a}(s_i)), G'(\mathbf{c}_{t=b}(s_i)))$. The terms M_i and K are used to normalise E_G relative to the contrast of the gradient of the curve, as some curves will naturally have larger image gradients along their paths than others.

With the curve locations at the key-frames I_a and I_b fixed, the Levenburg-Marquart algorithm is employed as the energy optimisation scheme to find the curve configurations in the between-key-frame frames with the lowest energy. This effectively propagates information both forwards and backwards through time, making the best use of the available user information. An example of tracked curves using this system is shown in Figure 5.12. By taking the image data into account, and encoding prior knowledge about how curves are allowed to deform, key-frames can be placed much further apart, reducing the amount of user interaction. The obvious caveat with this system is that if the image data is poor, due to motion blur, compression artefacts, object occlusion etc., the quality of the interpolated curves is likely to be poor also, and will require additional effort for correction.

5.3.2 Pixel based Cut out

The following pixel based video segmentation schemes represent the most relevant works in video object cut out, each having a clear motive in video post production applications. Each scheme is a logical extension of a 2D segmentation system, often treating the video as a 3D volume. The relevance of these systems to this thesis is the choice of feature space (or spaces) for calculating the data likelihood, and how user information is incorporated and propagated throughout the video.

5.3.2.1 Interactive Video Cut Out

Wang et al. extend the min-cut / max-flow framework of Grab-Cut to video by considering the video as a spatio-temporal volume [177]. One of the novelties of this approach is how the user supplied object and background scribbles are incorporated. The user is allowed to manipulate the 3D video volume, i.e. spin, rotate, pan, tilt etc. When the 2D scribbles are drawn on the volume, they can be projected through both time and space, allowing large portions of the video to be marked ("carved out") at once, as shown in Figure 5.13. The essence of the energy minimisation framework is the same as that of Grab-Cut, however each pixel site now has 4 spatial, and 2 temporal neighbours (excluding boundary pixels). It is generally agreed that the performance of any Graph-Cut is highly dependant on the number of the nodes[1]. The

[1]The edge connection arrangement of the nodes is also important when considering practical performance [27], but for the presented cases of 2D image or video segmentation, the pixel connections are assumed to be constant.

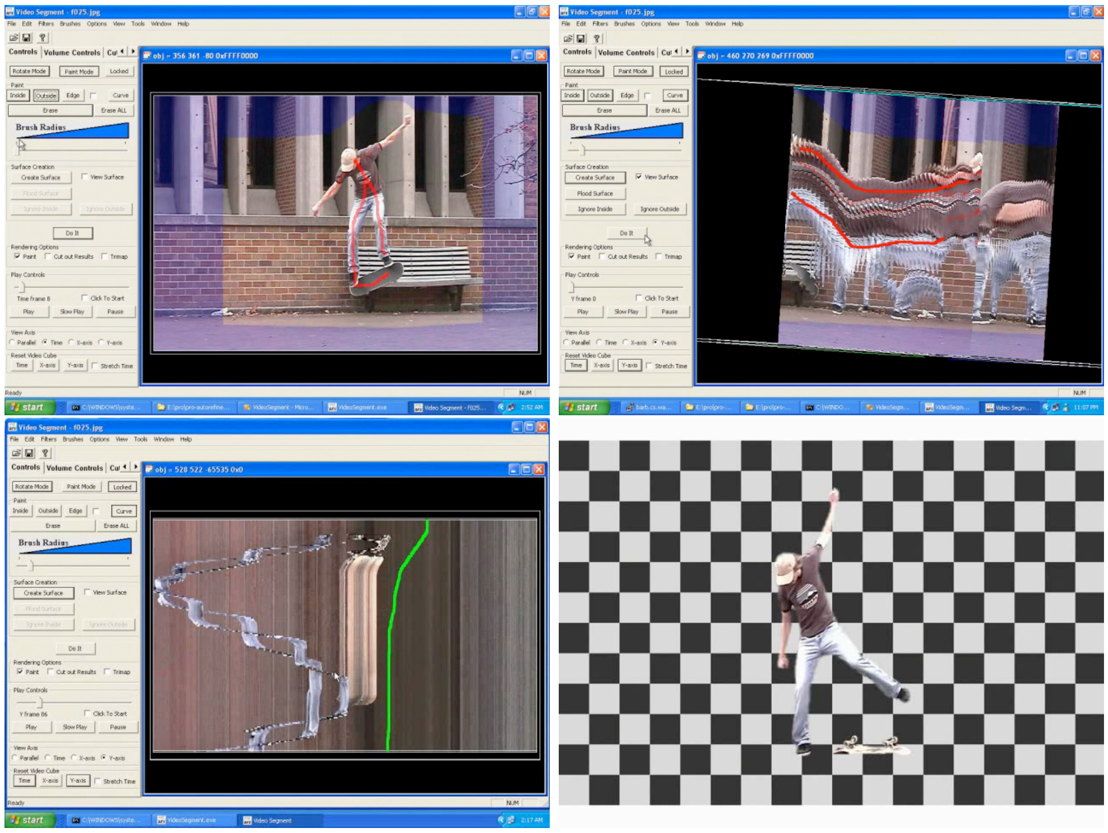


Figure 5.13: User interaction example of the "Interactive Video Cutout". The goal is to segment the skate-boarder from the background. Images from top left to bottom right. Initial user strokes are placed in the first image (red). The volumetric space is manipulated to give a novel space-time perspective, in which the skate-boarder can be selected through time, (red, 2nd image). Looking at a "slice" of video over time, large parts of the image known not to contain the object are marked as background (green, 3rd image). Last image, an accurate matte of the the skate-boarder is quickly extracted. (Images reproduced from Wang et al. [177].)

number of nodes in this case is the number of pixels in the video volume. Clearly, this means that Graph-Cut segmentation of the video in its volumetric form will be both slow and memory intensive. To try to remedy this, Wang et al. propose decomposing the volumetric representation into a hierarchical structure with three (fine to coarse) tiers; pixel level, 2D structure, and 3D spatio-temporal components. A course-to-fine (3D spatio-temporal volume to pixel level) segmentation is performed, allowing quick, interactive segmentation of videos. However, creation of the hierarchy takes a long time (10 - 30 minutes for 100 - 200 PAL resolution frames). In the system of Wang et al., a performance cost is incurred, either from using the generic Graph-Cut algorithm to perform the energy minimisation, or from creating the hierarchical video structure. However, representing and segmenting the video as a spatio-temporal volume at once is an interesting way of interacting with the system.

5.3.2.2 Distance-Cut for Video

Recall the use of geodesic distances for segmenting images. Similar to Wang et al., Bai & Sapiro extend a 2D system (in this case, their Distance-Cut system) to segment video by treating the video as a 3D volumetric space. The framework easily supports the transition from 2D to 3D; in the geodesic distance of Equation 5.6, ∇I now refers to the 3D gradient of the volumetric space I, and the curve C refers to the path between a pair of points in the 3D volume. Note that no motion compensation is performed between consecutive frames. To segment the object through time, the user first places scribbles in one frame to model the object and background colour distributions. Next, the object and background data likelihoods for each pixel in the video are used as weights in the 3D geodesic distance calculation, from the scribbles in the initial frame. This volumetric approach gives reasonable results for the first few frames, but as the geodesic distance starting points are taken from the scribbles, the system degenerates over time. For example, consider the case where a user scribble is placed on an object in frame 1 and by frame 3 the object has moved, such that the location of the scribble in frame 1 would now refer to the background in frame 3. It is conceivable that as the object likelihood is propagated forward in time from the scribble in frame 1, the geodesic distance to label the background region as object in frame 3 is less than the distance of labelling that same region as background, resulting in parts of the background being labelled as object. The rate of degeneration in the Distance-Cut system, and therefore how often corrections must be applied, is determined by the proximity (spatially and temporally) to the user scribbles, and how well separated the object and background colour distributions are. The system is clearly better suited for still images than for video, but the work of Bai & Sapiro has inspired renewed interest into using geodesic distances in video segmentation.

5.3.2.3 Geodesic Image Segmentation, "GeoS"

Improving on the Distance-Cut system, Criminisi et al. propose using the geodesic distance as an image filter instead [41], dubbed "GeoS". Morphological operations have been used extensively in the past for image segmentation [163, 112, 21]. For example, the non-parametric watershed algorithm partitions images based on how their level-sets are connected. Even simpler, using morphological closing operations on the results of a binary threshold of the segmentation likelihood as simple spatial consistency constraint. Criminisi et al. propose using the geodesic distance function to allow *real*-valued equivalents of the previously listed morphological operators. The interesting thing is that the new real-valued morphological operators are weighted by the image data. For example, a hole will only be filled if the gradient of the weighting around the boundary of the hole is sufficiently low. The GeoS framework is both intuitive and interesting, in the sense that there still exist simple and computationally inexpensive ways to achieve highly accurate segmentations. Criminisi et al. cast GeoS in the standard Bayesian energy minimisation framework of a data likelihood, U, and spatial prior, V.

Likelihood Like Bai & Sapiro, scribbles indicating object and background are supplied and used to build object and background colour distributions from which the likelihood is calculated. The foreground and background likelihoods, $P(I|\alpha = F)$ and $P(I|\alpha = B)$, are sampled directly from the histogram of colours (32 bins per channel) given by the user strokes.

The log likelihood ratio of the distributions is given by $L(\mathbf{x}) = \log\big(P(I|\alpha = F)/P(I|\alpha = B)\big)$. A sigmoid function is then applied to the log likelihood ratio, such that values in this transformed likelihood space are between 0 and 1. This is called the "log-odds map", $M(\mathbf{x}) = 1/\big(1 + \exp(-L(\mathbf{x})/\mu)\big)$.

Prior As with Bai & Sapiro, Criminisi et al. introduce spatial consistency by modifying the geodesic distance to incorporate a function of the likelihood distribution, M, as follows. The authors use a modification of the geodesic distance originally proposed by Toivanen [170] to allow a weighting, ν, between the log likelihood odds map and the geodesic distance,

$$D(\mathbf{x}; M, \nabla I) = \min_{\mathbf{x}' \in \Psi} \big(d(\mathbf{x}, \mathbf{x}') + \nu M(\mathbf{x}')\big) \tag{5.8}$$

$$d(\mathbf{a}, \mathbf{b}) = \min_{\boldsymbol{\Gamma} \in \mathcal{P}_{\mathbf{a},\mathbf{b}}} \int_0^1 \sqrt{\|\boldsymbol{\Gamma}'(s)\|^2 + \gamma^2(\nabla I \cdot \mathbf{u})^2} ds \tag{5.9}$$

where M is the "mask" image, Ψ is the 2D image space, ∇I is the 2D (or 3D) image gradient, $\mathcal{P}_{\mathbf{a},\mathbf{b}}$ is the set of all paths from points $\mathbf{a}$ to $\mathbf{b}$, $\boldsymbol{\Gamma}$ is one of these paths, parameterised by $s \in [0, \ldots, 1]$, $\boldsymbol{\Gamma}'$ is the derivative of $\partial\boldsymbol{\Gamma}(s)/\partial s$, and $\mathbf{u} = \boldsymbol{\Gamma}'(s)/\|\boldsymbol{\Gamma}'(s)\|$ is the unit tangent vector to the path. γ is used to weight between spatial (L^2) distance travelled and distance over image gradients.

Criminisi et al. generalise the "unsigned" geodesic distance to a "signed" distance as follows $D_s(\mathbf{x}; M, \nabla I) = D(\mathbf{x}; M, \nabla I) - D(\mathbf{x}; \overline{M}, \nabla I)$. Using the signed distance, image weighed dilation and erosion operators on the likelihood "log odds" map are given by $M_d(\mathbf{x}) = [D_s(\mathbf{x}; M, \nabla I) > \theta_d]$ and $M_e(\mathbf{x}) = [D_s(\mathbf{x}; M, \nabla I) > -\theta_e]$, for dilation and erosion thresholds θ_d and θ_e. The distance, D_s^s, is the result of symmetric dilation and erosion operation, defined as:

$$D_s^s(\mathbf{x}; M, \nabla I) = D(\mathbf{x}; M_e, \nabla I) - D(\mathbf{x}; \overline{M_d}, \nabla I) + \theta_d - \theta\ . \tag{5.10}$$

Manipulation of $D_s^s(\mathbf{x}; M, \nabla I)$ yields proposals of α by $[D_s^s(\mathbf{x}; M, \nabla I) > 0]$ for a given set of parameters $\theta = (\theta_d, \theta_e)$.

Optimisation Strategy Criminisi et al. propose to use a generalisation of geodesic distances to make proposals for α that are very close to a global minimum. The problem of doing this means choosing the best α labelling from analysis of the proposals, i.e. choosing α that maximises $P(\alpha|I, \theta)$. The α proposals come from evaluating $D_s^s(\mathbf{x}; M, \nabla I)$ for a range of parameters $\theta \in \mathcal{S}$, where for VGA sized images, Criminisi et al. set $\mathcal{S} = \{5, 6, \ldots, 15\} \times \{5, 6, \ldots, 15\}$. To find the best object labelling, the α proposals are supplied to the standard Bayesian energy function,

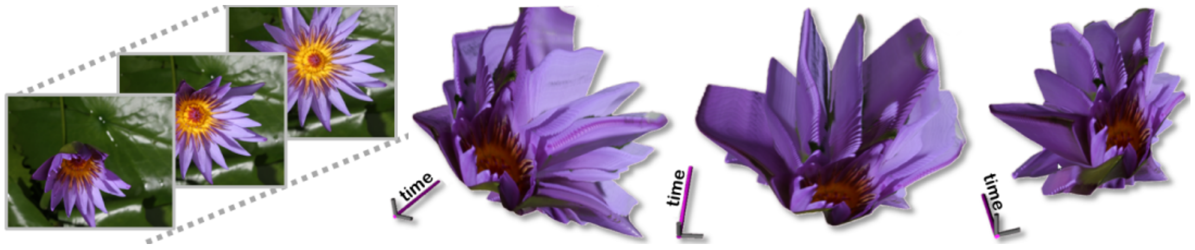


Figure 5.14: Example result of segmenting a flower through time in a sequence. First image, three example frames from the video sequence. Remaining images, views at various rotations of the segmented flower "volume", composed of the segmentation of each video frame over time. (Images reproduced from Criminisi et al.)

with the optimum $\hat{\alpha}$ labelling given by the optimum parameter set $\hat{\theta}$ with the lowest labelling cost,

$$\begin{aligned} \hat{\theta} &= \underset{\theta \in \mathcal{S}}{\operatorname{argmin}} \; E(\alpha, \theta, I) \\ \hat{\alpha} &= \alpha(\hat{\theta}) \; . \end{aligned}$$

By exploiting the near-linear time geodesic distance calculations, the segmentation is extremely fast, allowing rapid user interaction. As the information from the scribbles is purely to model the object and background colour distributions, Criminisi et al. show how it is possible to segment video, as shown in Figure 5.14, and n-D medical images from a few initial strokes. As video is by its nature a transient medium, the quality of the resultant segmentation depends on how much the object and background colours diverge from their models over time. However, given the fast, interactive speed of the system, the actual time required to correct (and inspect) the updated labelling results is significantly lower than previous systems.

5.3.2.4 Incorporating Motion

The video segmentation systems presented so far are the result of extending 2D segmentation algorithms to a 3D volume representation of video. For the segmentation of the object volume, it is assumed that consecutive pixels in time are connected the same way as spatially connected pixels, i.e. each pixel is simply connected to the 6 (or 26, including diagonal) neighbours in both space and time. This makes sense in the 2D (or actual 3D images, such as medical) where the assumption of spatial smoothness is reasonable, however it is well known that video often exhibits local discontinuities in time, for example due to foreground or background occlusions. However, none of the presented systems properly address how pixels in one frame relate to pixels in the next frame, which often results in errors in the cut out, as evident by the degeneration of the Distance-Cut system over time. The obvious way to introduce better temporal coherency is look at the apparent motion between frames.

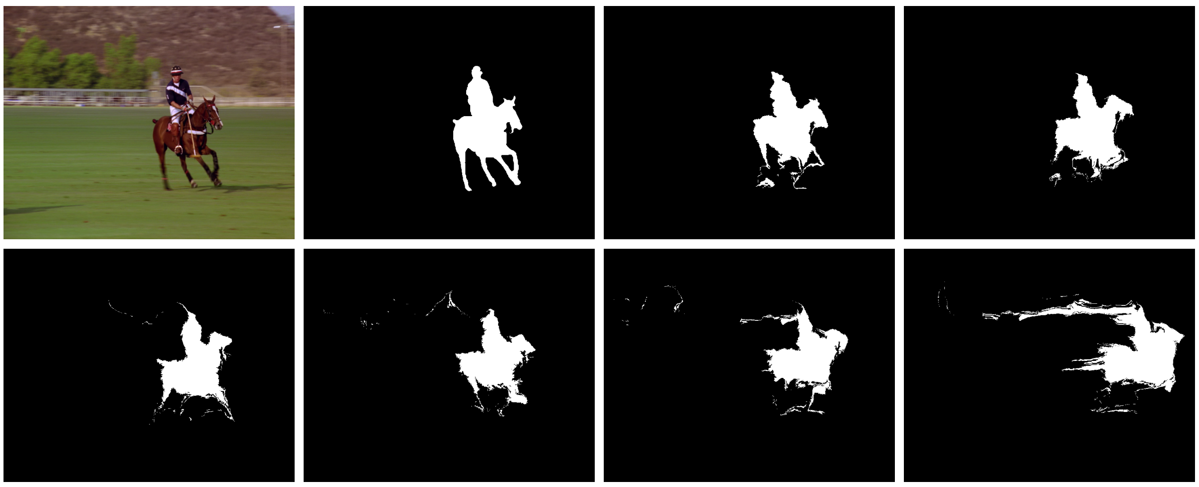


Figure 5.15: Example of propagating a user matte across 60 frames using motion vectors alone. The original image and corresponding matte from the "Polo" sequence are shown (top, far-left) and (top, centre-left) respectively. The matte is then "pulled" through the video sequence, where each new label is given by the previous frame label advected by the calculated motion vectors. For example, the matte is pulled to frame 2 from frame 1, the resultant matte is then pulled to frame 3 from frame 2, then to frame 4 from frame 3, and so on. The results for frames 10, 20, 30, 40, 50 and 60 are shown by the (top, centre-right) to (bottom, far-left) images. Notice that although the body of the horse and rider remains relatively intact, difficult motion regions degrade quickly over time, for example, the wind-swept appearance of the horse's legs and rider's head. Although the resultant matte for frame 60 is not great, there is still some useful information there, indicating that motion is a useful feature space.

A large strand of video segmentation research uses motion to remove the user from the workflow, automating the entire process. In this way, motion vector fields are used as an additional feature space in a probabilistic framework, as shown in [84], [82], [19], [180], [12] and [40]. Given a user created object cut out in one frame, it is logical to use the apparent motion to propagate the mask to neighbouring frames, as shown by Gu & Lee [68], and Choi et al. [33]. An example of motion based cut out propagation is shown in Figure 5.15. Apostoloff & Fitzgibbon [7, 8] propose a semi-automatic segmentation system based on identifying occlusion boundaries given by "T" junctions in transverse slices of the spatio-temporal video volume. The detected occlusion boundaries are then used to define an additional pair-wise spatial energy term, where the idea is to reduce the cost of assigning neighbouring pixels to different α labels if they are situated near an occlusion boundary. This intuitive contribution to the standard energy function makes use of the temporal behaviour of objects common in most video footage, and deserves further study.

Kokaram et al. present an interesting approach to video object segmentation [84], by casting the problem as a "blotch" detection problem. The idea is that the object and background often have different apparent motion models, and although the object may deform in interesting

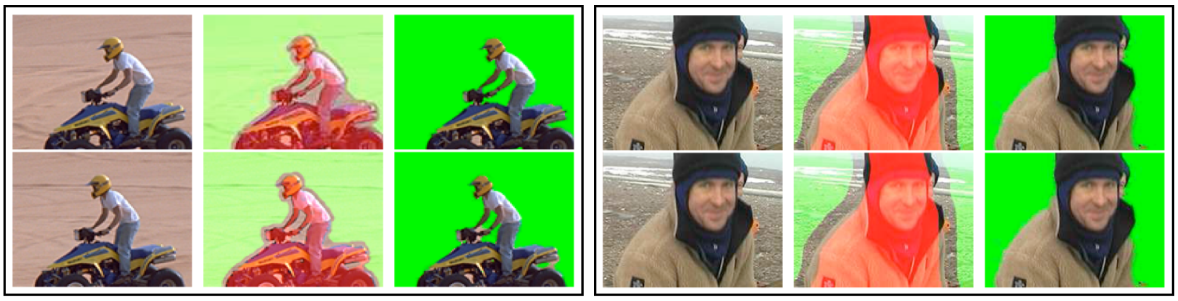


Figure 5.16: Example of segmenting objects by apparent motion layers, on two sequences (left and right groups). In each group; first column, original "natural" video frames. Second column, example of "garbage matte" generated from the separation of motion layers, where the green and red areas indicate confident background and foreground respectively. Last column, the garbage matte is used in a non-binary matting algorithm to give a higher quality matte. (Images courtesy of Kokaram et al. [84])

and unknown ways, the background will generally move consistently. By modeling the global motion in the scene (for example, by homographic transform estimated by optic flow between frames), pixels that do not obey the motion model have a higher likelihood of belonging to the object. This likelihood is used in the standard Bayesian segmentation framework (as described earlier), resulting in a "garbage matte", which can then be used as input to a non-binary matting algorithm to give a sharper, higher quality matte. An example is shown in Figure 5.16. This is sometimes referred to as *layer-based* segmentation [182], where it is assumed that the video sequence can be decomposed into multiple layers based on apparent motion. Wills et al. propose using feature points to both estimate the motion in a scene, and fit the appropriate motion models [182]. The idea of incorporating motion to improve video segmentation is interesting, in that the temporal nature of the video is now being properly considered. However, the success of these algorithms often relies heavily on the accuracy and reliability of the motion vectors. If the motion estimation process fails, the automated segmentation process fails too, and the user is back to manual correction.

5.3.2.5 Combining Multiple Cues

With the exception of motion based cut out systems, the majority of pixel-based segmentation algorithms use colour as a dominant feature space to model the object and background likelihoods. Although colour can be very useful, it is unreasonable to assume that object and background colour distributions remain sufficiently constant over the duration of a video. In addition, the chosen modelling scheme (such as GMM or KDE) may not accurately represent the colour distributions, or the actual foreground and background distributions simply may not be separable. In these cases, colour based image or video segmentation will fail, and the user is required to correct the resulting cut out manually. Recent research has looked at how to

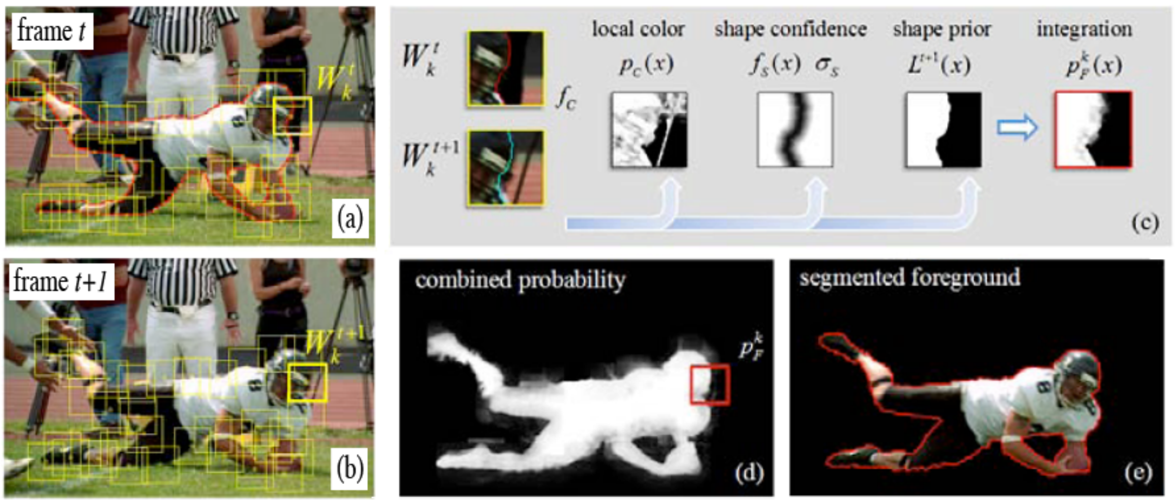


Figure 5.17: An example of how Video Snapcut's per-region object likelihood is calculated by balancing shape and colour information to give an accurate segmentation. Top left (a), shows examples of the regions W_k^t sampled along the manually specified object boundary (red line) in the initial frame t. The corresponding regions are then tracked and found again in a later frame $t+1$, bottom left (b). Top right (c), a pair of corresponding regions selected from consecutive frames are selected for comparison. The object colour likelihood $p_c(x)$ for frame $t+1$ is calculated using a colour distributions modelled in manual frame t. The shape confidence $f_s(x)\sigma_s$ models how well colour can separate foreground and background around the boundary region. The width of the ambiguous region around the boundary is given by σ_s, which is inversely proportional to the colour distribution separation ("colour confidence"). The shape prior is given simply by the pulling of the user labelling from frame t to $t+1$ in the regions W_k^t and W_k^{t+1}. The object likelihood for region W_k^{t+1} is the blend of colour and shape information, using the shape confidence to select the appropriate cue weighting for each pixel. (Images courtesy of Bai et al.)

incorporate multiple feature cues to achieve more reliable segmentations over longer sequences with less user interaction.

Video Snap-Cut Bai et al. [11] propose a video segmentation scheme known as "Video Snap-Cut", which exploits motion, colour and shape cues. The idea is to first manually label the object in an initial video frame, giving the system a complete example of the object. Overlapping regions are then sampled uniformly along the boundary contour of the object, where each of the regions W_k is likely to contain both foreground and background. These regions are then tracked through subsequent video frames. Colour and boundary shape information from the manually segmented frame are then propagated forward into the tracked regions, and used to segment the object in later unmarked frames. The novelty of Bai et al. over simply pulling the manual cut out into new frames using estimated motion vectors alone is two-fold. Firstly, the tracking stage is not using the manual cut out directly, rather object statistics (colour and shape) from the *local* cut out regions are used to calculate the likelihoods in the corresponding regions in the

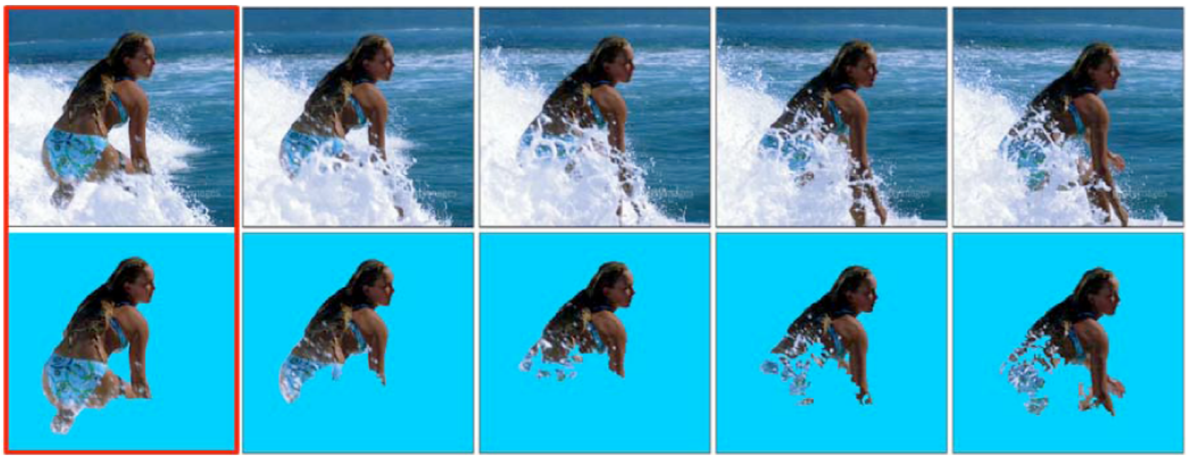


Figure 5.18: Example of results obtained by the Video Snap-cut system of Bai et al. The object is first manually segmented in a key-frame (red). The surfer is reasonably well cut out in the presence of difficult occlusions due to the surf spray. Also notice that the system handles change in object topology well, for example the hand correctly segmented in the last frame despite not appearing in the original key-frame. (Images courtesy of Bai et al.)

new frame. This implicitly allows for reasonable errors in the motion estimation. Secondly, Bai et al. propose an adaptive weighting between colour and boundary shape for each region W_k. Using the colour information of a region W_k in a manually segmented frame, the separability of the object and background in the proposed region in the new frame is measured as a "colour confidence". If the colour distributions are clearly separable (a high "colour confidence"), then colour information is probably a useful cue for segmentation in this region, and is weighted higher than the boundary shape information. However, if the colour confidence is low, for example in a region W_k where foreground and background share similar colours, then the shape of the boundary from W_k is a more useful feature for segmenting the object, and is weighed higher accordingly. An example of how the colour and shape information are combined is shown in Figure 5.17. The object cut out in the unmarked frame is then found by summing the likelihoods for all the regions. Given that the regions overlap, information from many propagated regions can contribute to the data likelihood, shown in Fig. 5.17 (d). This likelihood is used directly in the standard Graph-Cut energy minimisation scheme of Equation 5.2 to give an initial object cut out for that frame. The authors highlight the fact that it is only the regions W_k in which the labelling needs to be solved for, and not the entire image. This allows for iteratively solving for both the optimum segmentation and model parameters similar to that of Grab-Cut [148], with the majority of regions converging within 3 iterations. A non-binary matting process is then applied to the resultant cut out. An example of a difficult cut out using Video Snap-cut is shown in Figure 5.18.

LIVEcut Other impressive recent work in video segmentation by Price et al. [136] also attempt to intuitively combine information from multiple cues. Similar to Bai et al., the user first

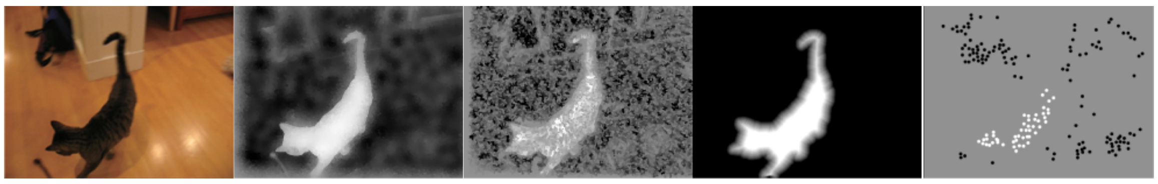

(a) Original (b) Colour (c) Temporal (d) Shape (e) Tracking

Figure 5.19: Example of feature cues used in the LIVEcut segmentation system. From left to right, Original frame, Colour likelihood w_c, Spatio-temporal coherence w_h, Shape w_s, and Point tracking w_p. White indicates a high probability of belonging to the object. The idea is to exploit the agreement between many cues to mitigate errors typically exhibited by using single cues alone. For example, the rope in the bottom left of the frame has a high colour likelihood of being classed as object as it shares the same colour as the cat. However the rope has a low likelihood in every other feature space, and so will probably not be classed as object. (Images courtesy of Price et al.)

marks the entire object manually in one frame. In subsequent unmarked frames, the "LIVEcut" system calculates object likelihoods based on multiple feature cues, such as colour, shape, spatio-temporal coherence and point tracking. The novelty of this system is the way in which many simple cues are weighted and combined to form the data likelihood.

Likelihood Unlike the Video Snap-Cut system of Bai et al. [11], where only two cues (colour and shape) are considered following a motion compensation step, the LIVEcut system provides a tidy framework for combining an arbitrary number of cues, and weighting each cue based on how well the cue is able to segment the object, given the user's manual cut outs or corrections. The data energy term is defined as follows,

$$U(\alpha, I, \mathbf{x}) = s(\mathbf{x}, \alpha) + \sum_{u \in \mathcal{U}} a_u(\mathbf{x}) w_u(\alpha, I, \mathbf{x})$$

where U is the set of feature cues, $w_u(\alpha, I, \mathbf{x})$ is a function of the feature cue $u \in \mathcal{U}$, giving the cost of assigning a pixel at site $\mathbf{x}$ to label α. The term $a(\mathbf{x})$ is a per-pixel scalar weighting the contribution of cue u to the overall data energy. To incorporate any manual corrections, the term $s(\mathbf{x}, \alpha)$ is set to ∞ for pixels the user has labelled as foreground given $\alpha = B$, or labelled background given $\alpha = F$, and set to 0 otherwise.

The four feature cues used by LIVEcut are colour w_c, spatio-temporal coherency w_h, shape w_s, and point tracking w_p. The colour term is calculated in the standard manner of modelling the colour distributions using the manually supplied object and background mattes. The shape term is generated by warping a user defined matte from a previous frame into the current frame (using a homography calculated from feature correspondences), reducing the effects around the boundary of the propagated matte to allow for non-rigid deformations between frames. The spatio-temporal consistency term is given simply by the DFD between motion compensated frames. Lastly, the point tracking term follows points from the user supplied matte frames into the current frame using the KLT tracker [156, 171], and uses these points as hard likelihood "seeds". Pixels close to the tracked points in the current frame are encouraged to have the same label as the regions from which the points came from in the original user labelled frame. Examples of the cues on a real image shown in Figure 5.19. The novel methods of weighting the various cues are now presented.

The weighting of each cue in the energy function, $a_u(\mathbf{x})$, is a product of automatic β and semi-automatic terms ρ, i.e. $a_u(\mathbf{x}) = \beta_u(I, \mathbf{x})\rho(I, \mathbf{x})$. The term $\beta_u(I, \mathbf{x})$ is essentially an automatic validation term, which uses the model derived from the manually segmented frame to measure how useful a given cue is at extracting the object for that already segmented frame. Taking the colour cue for example, given a manual labelling of a frame $\mathcal{L}$, and modelled colour distribution data θ, the automatic weighting is calculated as $\beta_c(I, \mathbf{x}) = P(I|\mathcal{L}, \theta, \mathbf{x})$. The weighting $\beta_c(I, \mathbf{x})$ will therefore have high values in regions where colour is a useful feature for segmenting the object, and low values otherwise. Similar validation functions are calculated for the remaining cues. This idea is similar to the automatic weighting between colour and shape information of the Video Snap-cut system. However, the LIVEcut system allows independent weighting of many cues, and more importantly, the weighting $a_u(\mathbf{x})$ is calculated *per pixel*, instead of over the whole image (or in the case of Video Snap-cut, only near the object boundary in the local region W_k), allowing greater flexibility in selecting the best feature spaces to use.

Another novelty of LIVEcut is the scheme used to weight the local contributions of feature cues based on user corrections. The idea is to see where the user is making corrections as the video segmentation is proceeding, and adjust the weighting of the cues in that region to improve the automatic segmentation of future frames. At the corrected pixel sites, each of the cues are tested, measuring how well each cue could capture the correction. For example, consider the case where a part of the background, with the same colour as the object, has been selected as belonging to the object. The user then corrects the area, and the segmentation is performed again for that frame. Assume that S_u^i is a putative segmentation given simply by testing the cue u on its own for site $\mathbf{x}$, i.e. $[w_u(F, I, \mathbf{x}) < w_u(B, I, \mathbf{x})]$, and that S_u^f is the final segmentation of the system following corrections. For each cue, $w_u(I, \mathbf{x}, \alpha)$, the user correction weighting is weighted down for those sites in subsequent frames where $S_u^i \neq S_u^f$, and weighted higher otherwise, i.e.

$$\rho_u^{\text{next}}(\mathbf{x}) = \begin{cases} \rho_u(\mathbf{x}) + \delta_0 & \text{if } S_u^i(\mathbf{x}) = S_u^f(\mathbf{x}) \\ \rho_u(\mathbf{x}) - \delta_1 & \text{if } S_u^i(\mathbf{x}) \neq S_u^f(\mathbf{x}) \end{cases}$$

where $\rho_u^{\text{next}}(\mathbf{x})$ is the updated user correction weighting to be used in later frames, and δ_0 and δ_1 are update parameters, set to 0.4 and 0.8 respectively. $\rho_u(\mathbf{x})$ is initialised to 1 at the start of the overall segmentation process.

Prior The spatial energy term of Price et al. is similar to other spatial prior terms, in that spatial smoothness of the label field is encouraged in smooth areas, and allowed to differ more between pixel neighbours with high gradients. However, Price et al. also include an additional

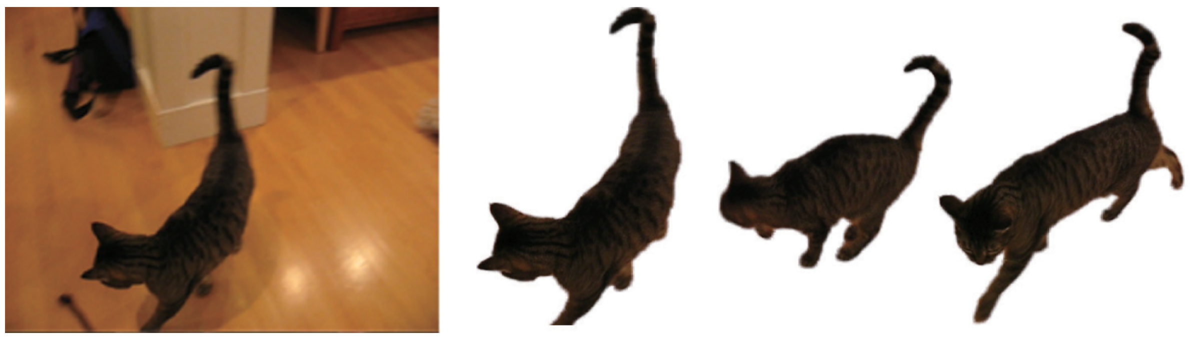


Figure 5.20: Example of results obtained by the LIVEcut system of Price et al. Three segmented example frames are shown from the “Cats Playing” video. This sequence is difficult for many reasons, such as objects in the same scene with similar colour (the rope, the other cat, and the background shadows), multiple complicated motions (from both cats and camera motion), and large changes in object topology (the cat’s legs, tail and ears). These difficulties would usually cause traditional systems to fail or require heavy corrections. However, the high quality segmentation results obtained by the LIVEcut system show the effects of combining many complimentary feature cues effectively. (Images courtesy of Price et al.)

colour adjacency term to further improve the label consistency. The idea behind colour adjacency, is that certain colour pairs will only ever occur entirely within the object, or within the background, and never along the boundary. The colour adjacency term, w_a, is given by modelling the colour distributions between neighbouring pixels. Two distributions are generated, one for the interior regions of the object *or* background, and another for the boundary sites between object and background, as defined by the previously labelled frame $\mathcal{L}$. In the new frame, the edge cost for assigning two neighbouring pixels different labels is reduced if the edge is likely to belong to a border pixel, estimated by the joint probability of observing the colours of the pixel pair given the modelled distributions. Including a simple gradient cost, the LIVEcut spatial energy $V(\alpha, I, \mathbf{C})$ is then defined as follows,

$$V(\alpha, I, \mathbf{C}) = \sum_{(\mathbf{x}_i, \mathbf{x}_j)\in\mathbf{C}} w_a(I, \mathbf{x}_i, \mathbf{x}_j, \mathbf{C}) + \sum_{(\mathbf{x}_i, \mathbf{x}_j)\in\mathbf{C}} \frac{1}{\|I(\mathbf{x}_i) - I(\mathbf{x}_j)\|^2 + 1}$$

where w_a is the colour adjacency term, and $\mathbf{C}$ is the set of pixel cliques in the image I.

Optimisation To solve for the optimum labelling, $\hat{\alpha} = \underset{\alpha}{\text{argmin}}\ E(\alpha, \theta, I)$, the data and spatial energy terms, $U(\alpha, I, \mathbf{x})$ and $V(\alpha, I, \mathbf{C})$ are used in the standard Graph-Cut minimisation scheme of Boykov & Jolly [25]. However, an interesting thing to note is that the combination of multiple complimentary cues in the LIVEcut system allows for a single, “one-shot” Graph-Cut operation (as opposed to the ping pong technique of Grab-Cut), improving computational efficiency without sacrificing segmentation accuracy. A brief example of results highlighting the strengths of the LIVEcut system are shown in Figure 5.20. As with other segmentation systems intended for post production use, a subsequent matting stage is applied to obtain a smooth

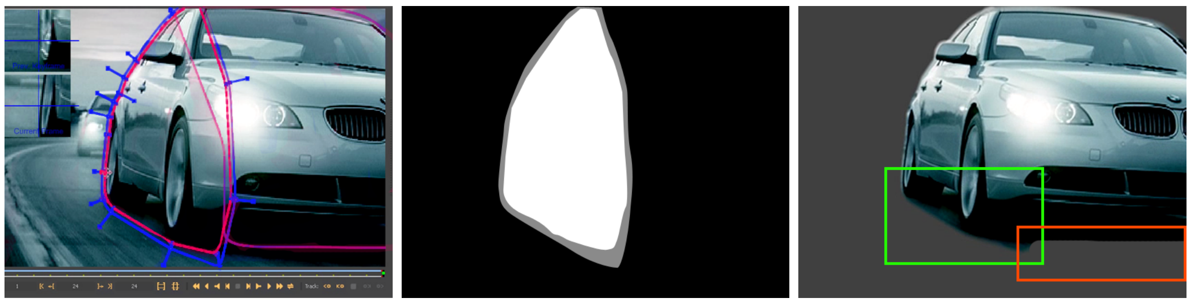


Figure 5.21: Example of matting using rotoscope curves in Imagineer System's "Motor" BMW sequence, the goal is to create a smooth segmentation around the shadow and the road. Left, an ambiguous area is specified using two curves (inside and outside, red and blue respectively). This is known as "feathering". Centre, a "tri-map" is calculated as the difference between one of the two roto curve pairs, where definite object is shown in white, definite background in black, and the "unknown" region in which the non-binary labelling is to be calculated, shown in grey. Right, the effect of the non-binary label assignment process is shown in the green square. The red square shows the cut out results in the absence of feathering. Notice how the feathering produces a soft, non-binary edge along the shadow of the car, giving a more realistic look than the binary cut out shown in the red square. (Images courtesy of Imagineer Systems.)

non-binary α labelling.

5.4 Matting

In matting literature, it is assumed that the colour of a pixel in the image, C, is a function of the foreground (object) colour, F, and the background colour, B, given by the *compositing* equation [35]:

$$C = \alpha F + (1 - \alpha)B \tag{5.11}$$

where C is the resultant pixel colour (minus camera noise, σ_c), so if a pixel has three colour components, i.e. RGB, YCbCr, HUV, Lab etc., then for each pixel in the unknown region, there are 3 equations with 7 unknowns to solve for (F, B & α). A common matting task involves estimating the non-binary, α values of object and background following a successful (or even near successful) object cut out. Another task relates to pathological content, in which some of the scene content is more difficult to define as object or background, such as smoke, glass, or other semi-opaque entities. For the purposes of this discussion on post production object cut out, the matting scenario is constrained to non-pathological content, where the image is decomposed into a definite foreground region, a definite background region, and an unknown region containing some linear mix between the two. This separation image is known as a "tri-map", and is generated by both pixel- and spline based systems, an example using roto curves is shown in Figure 5.21. As noted by Levin et al., most recent approaches to matting involve

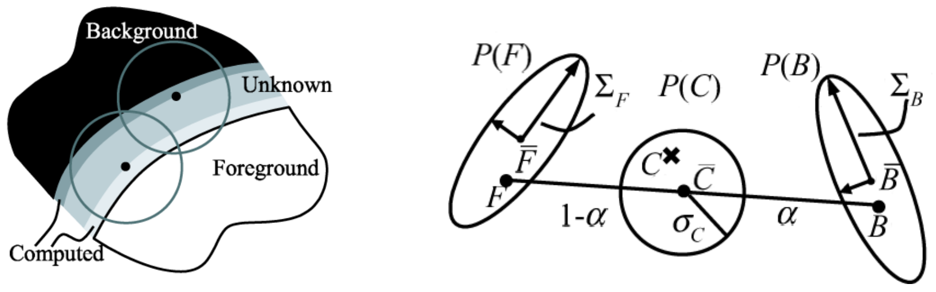


Figure 5.22: Example of Bayesian matting neighbourhood regions (left), and colour model (right). (Images reproduced from [35])

the use of a tri-map [35, 181, 140, 78] to solve this "severely underconstrained" problem (i.e. 3 equations, 7 unknowns per pixel) [92].

Although this thesis is not directly concerned with matting, it is an essential tool in post production, with a large volume of impressive research behind it [141, 6, 66, 181, 140, 179]. As such, two interesting matting algorithms are selected for discussion from the comprehensive comparative study of Rhemann et al. [141]; "Bayesian Matting" [35], and "Closed-form Matting" [92]. These two papers are chosen as their solutions are elegant and provide clear insight into the difficult matting problem.

5.4.1 Bayesian Matting

The work of Chuang et al. apply a Bayesian framework to matting [35]. In order to solve for α, F, and B at a pixel site, the *calculated* colour distributions of F and B in the unknown region are used as well as those marked as foreground and background regions in the tri-map. The problem is set up as a maximum a posteriori (MAP) problem to find the most likely values for α, F and B given the observation C,

$$\begin{aligned}\underset{\alpha,F,B}{\operatorname{argmax}}\ P(F,B,\alpha|C) &= \underset{\alpha,F,B}{\operatorname{argmax}}\ P(C|F,B,\alpha)P(F)P(B)P(\alpha)/P(C) \\ &= \underset{\alpha,F,B}{\operatorname{argmax}}\ L(C|F,B,\alpha)+L(F)+L(B)+L(\alpha)\end{aligned} \tag{5.12}$$

where $L(.)$ denotes the log likelihood operator ($\log P(\cdot)$), and the $P(C)$ term is dropped as it is constant with respect to the optimisation. Chuang et al. model the first term, $L(C|F,B,\alpha)$, as the difference between the observed colour C and the predicted colour given by estimates of F, B and α:

$$L(C|F,B,\alpha) = -\|C-\alpha F-(1-\alpha)B\|^2/\sigma_C^2\ . \tag{5.13}$$

This models the measurement error in C, and is modelled as a Gaussian probability centred at $\bar{C} = \alpha F+(1-\alpha)B$ with standard deviation σ_C, as shown in Figure 5.22 (right). The foreground colour distribution $P(F)$ is built using colour values in the pixel's neighbourhood N, shown by

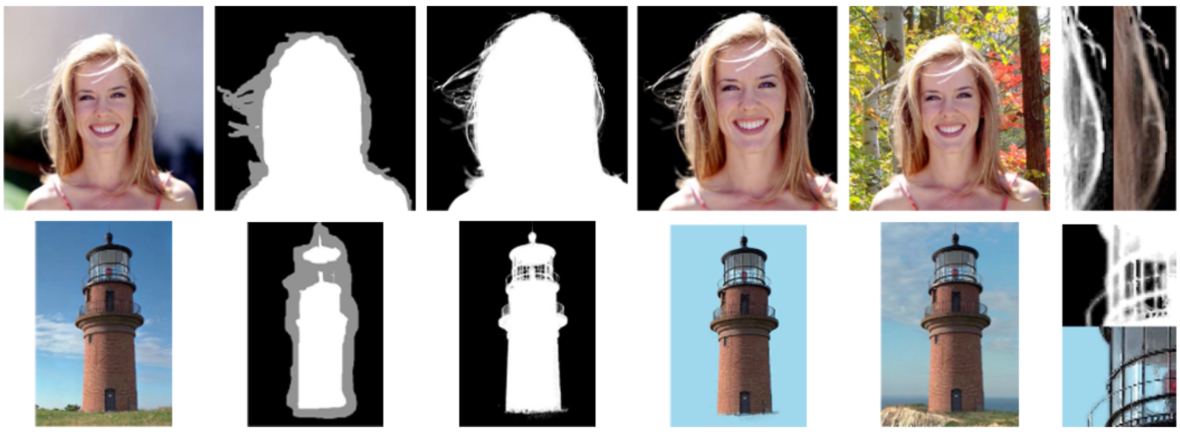


Figure 5.23: Example of "natural image" mattes produced by the Bayesian Matting framework of Chuang et al. Columns, from left to right; Original image, user defined "tri-map", alpha matte produced by program, composite onto uniform colour, composite onto novel background, matte close-up. Notice how the system accurately captures the wisps of hair (top row, last image), and the semi-opaque windows and railings, as well as providing "soft" borders to the object boundaries (bottom row, last image). (Images reproduced from Chuang et al. [35])

the circles in Figure 5.22 (left). Samples of the foreground colour F in N are added to the distribution weighted by $w_i = \alpha_i^2 g_i$, thereby favouring more opaque pixels, that are also close to the centre of the Gaussian window function g. The distribution is then modelled by the weighted mean, $\bar{F} = \frac{1}{W}\sum_{i\in N} w_i F_i$, and covariance matrix $\Sigma_F = \frac{1}{W}\sum_{i\in N} w_i(F_i - \bar{F})(F_i - \bar{F})^T$, where $W = \sum_{i\in N} w_i$. The log likelihood for a pixel having foreground colour F is given simply by the log of the multi-dimensional Gaussian probability of observing F:

$$L(F) = -(F - \bar{F})^T \Sigma_F^{-1}(F - \bar{F})/\sigma_C \ .$$

Chuang et al. note that the definition of $L(B)$ will be different depending on the matting task, such as "constant colour matting", where the mean and covariance of the background colour is sampled strictly from labelled pixels of the object cut out, or "difference matting", where the background colour distribution is known in advance (blue or green screen, or the background is fixed, i.e. for background subtraction). The task of "natural matting" is considered here, and so the distribution for $P(B)$ is sampled and modelled similarly to $P(F)$. The relationship between $P(F)$, $P(B)$ and $\bar{C}$ is illustrated in Figure 5.22 (right).

To solve for F, B and α, Chuang et al. break the problem into two sub-problems, as the multiplications of α with F and B in the log likelihood $L(C|F,B,\alpha)$ in Equation 5.13 means the optimisation of Equation 5.13 is not quadratic with respect to α, F and B. The first sub-problem involves solving for F and B by keeping α constant. The partial derivatives of Equation 5.13 with respect to F and B are set to 0, resulting in:

$$\begin{bmatrix} \Sigma_F^{-1} + I\alpha^2/\sigma_C^2 & I\alpha(1-\alpha)/\sigma^2/\sigma_C^2 \\ I\alpha(1-\alpha)/\sigma_C^2 & \Sigma_B^{-1} + I(1-\alpha)^2/\sigma_C^2 \end{bmatrix} \begin{bmatrix} F \\ B \end{bmatrix} = \begin{bmatrix} \Sigma_F^{-1}\overline{F} + C\alpha/\sigma_C^2 \\ \Sigma_B^{-1}\overline{B} + C(1-\alpha)/\sigma_C^2 \end{bmatrix} \tag{5.14}$$

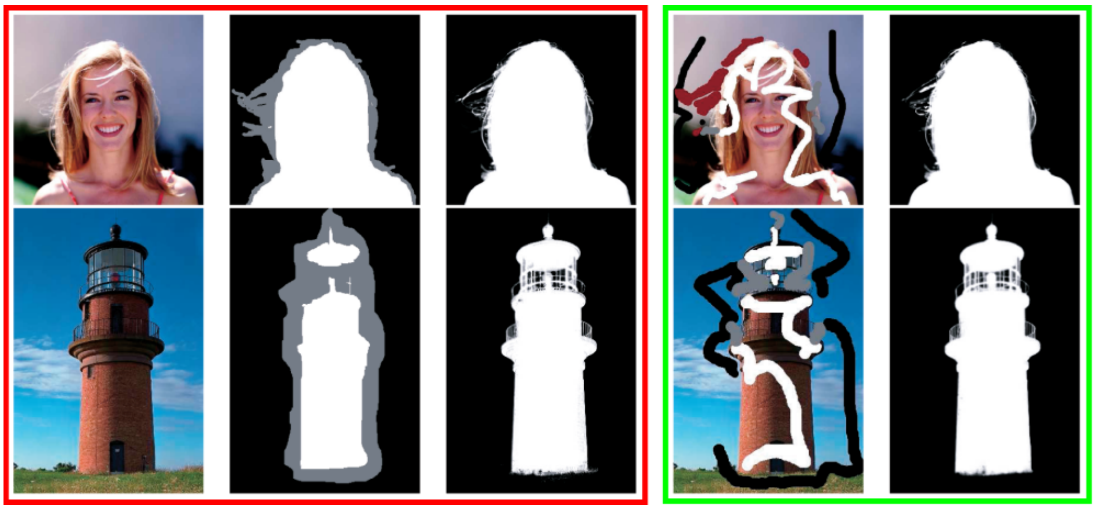


Figure 5.24: Comparison of "natural image" mattes, using images from the Bayesian matting section in Figure 5.23 (red rectangle) versus the same images in Closed-Form matting (green rectangle). Columns, from left to right; original images, tri-map used in Bayesian matting, Bayesian α matte, user-scribbles for Closed-Form matting, Closed-Form α matte. Notice that although the two approaches produce very similar mattes, there are some strange artefacts in the Bayesian mattes, for example, on the long horizontal wisp of the woman's hair, and at the base, right, of the lighthouse. (Images reproduced from Chuang et al. [35] and Levin et al. [92])

where I is the 3×3 identity matrix. The optimum values for F and B are given by the solution of Equation 5.14 for a constant α. The second sub-problem assumes that F and B are constant and solves for α, by observing that C lies somewhere (α) along the line between F and B in colour space (from Figure 5.22 (right)),

$$\alpha = \frac{(C - B) \cdot (F - B)}{\|F - B\|^2} \,. \tag{5.15}$$

The Equation 5.13 is optimised to find the values of α, F and B, by alternating between Equations 5.14 and 5.15 until convergence. The initial α values are seeded by the mean α values in the neighbourhood (using $\alpha = 1$ and $\alpha = 0$ from the hard foreground and background labelled regions for example), before solving the sub-problem of Equation 5.14. The authors note that this model uses only one local foreground and background colour model. Although it is possible to conceive situations where F or B should be modelled by more than one model (i.e. where two distinct colours are spatially very close), the results of the matting are still of a high quality. Some example results of natural image matting using the Bayesian matting approach are shown in Figure 5.23.

5.4.2 Closed-Form Matting

Levin et al. [92] propose a novel approach to natural image matting. As the system is complicated, only the core contributions of the paper are presented. The authors first rewrite the compositing equation (Eq. 5.11) to make α linear with respect to the image data,

$$\alpha = aI + b \tag{5.16}$$

where $a = \frac{1}{F-B}$, $b = \frac{B}{F-B}$, and F & B are the foreground and background colours. This rearrangement of the compositing equation is to simplify the computation of α by enabling the terms a and b to be eliminated. The idea is to assume that for small image blocks, a and b are fixed, so for a given α it is possible to derive an expression to find the best values of a and b, denoted $J(\alpha, a, b)$. Because a and b are constrained over the local image block, it is possible to analytically eliminate a and b through marginalisation, resulting in a quadratic cost function in α,

$$J(\alpha) = \alpha^T L \alpha \tag{5.17}$$

where L is the sparse, pixel connectivity matrix of size $N \times N$ (where N is the number of image pixels) created through the marginalisation process of $J(\alpha) = \min_{a,b} J(\alpha, a, b)$. Each (i, j)th entry in the matrix L is based on the difference in colours between pixels i and j. Intuitively, L is similar to a Laplacian matrix, and is used to drive the diffusion of α throughout the image. To supply user information, Levin et al. constrain the α values in Equation 5.17 for regions belonging to definite foreground or background regions specified by a tri-map or user scribbles. As Equation 5.17 is quadratic with respect to α, the optimum $\hat{\alpha}$ is solved using the least-squares solution of the sparse linear system. This quadratic form is typically found in partial differential equation (PDE) systems. Using the calculated α label field, F and B can then by reconstructed by a least-squares estimate using the compositing equation (Eq. 5.11) and encouraging local spatial smoothness. Results of the closed-form matting technique of Levin et al. are superior to those of the Bayesian matting approach of Chuang et al., examples of which are shown in Figures 5.24 and 5.25, with their Bayesian counterparts included for visual comparison.

Levin et al. develop the user constraints further, such as fixing the foreground or background colours under the scribbles and allowing user input to handle pathological content, however that is outside the scope of this introductory matting section. As well as the one-shot approach to solving the matting problem, one of the attractive things about the closed-form matting approach is the diagnostic information given by the connectivity matrix L. By a slight modification of how the elements of the matrix L are computed, it is possible to exploit eigenvector analysis techniques from spectral segmentation literature (such as [155] or [66]) to find the optimum regions where the user should place the strokes for maximum effect. Levin et al. note that the connectivity matrix L already contains a lot of useful information before any user information is supplied. The idea is that coherent regions in the the calculated alpha matte correspond to coherent regions in the lowest eigenvectors. This means that by placing user strokes in the regions corresponding to the coherent regions in the eigenvector image, the user stroke will have the maximum effect, having the furthest reaching propagation. An example is shown in Figure 5.26[2].

[2]In a sense, this is similar to a geodesic distance between a specified point in the image colour space, however

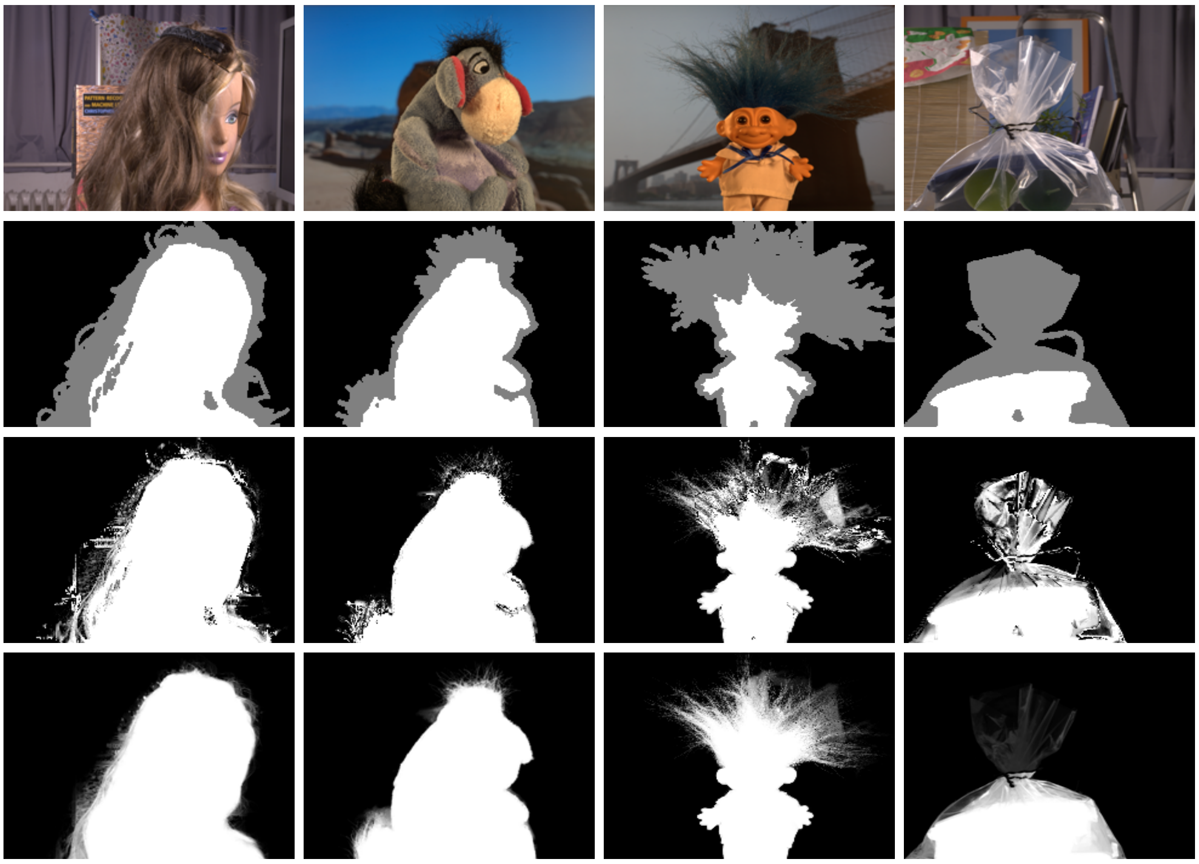


Figure 5.25: A comparison of mattes on more difficult "natural images". Top row, original images. 2nd row, user supplied tri-maps. 3rd row, results from the Bayesian matting approach of Chuang et al. Last row, results of the Closed-Form matting approach of Levin et al. Notice that although the Bayesian matting produces mattes that are reasonable, the Closed-Form solution produces mattes of a considerably higher quality, even on difficult images such as the plastic bag (right column). (Images courtesy of Rhemann et al. [141], http://www.alphamatting.com)

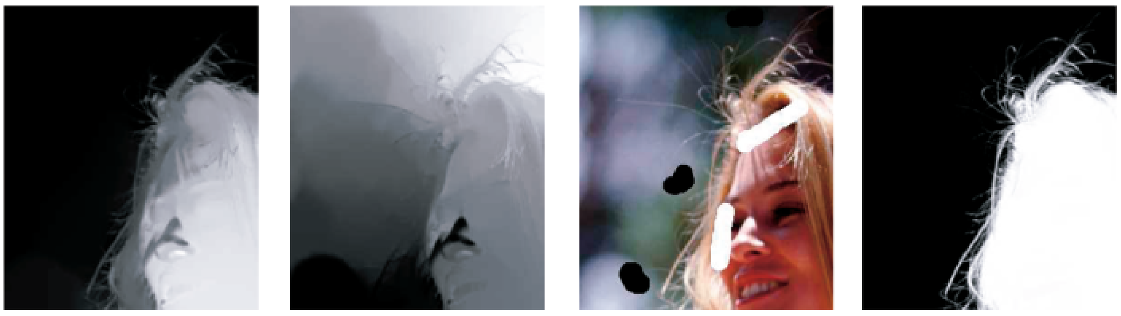


Figure 5.26: Analysis of the first two smallest eigenvectors (first two images) yield the optimal placement for user background and foreground scribbles. Regions with similar grey level values are considered connected. In the third image, scribbles are drawn in the "coherent regions" (of smooth appearance) in the eigenvector images. Last image, the resulting matte of only using eigenvector analysis to decide scribble placement. (Images reproduced from [92])

5.5 Discussion & Conclusion

Regardless of the cut out or matting system, near-perfect results can be obtained by the explicit manual cut out of each frame. That is the upper bound on both time and cut out quality. With this in mind, the objective of any system is to reduce the amount of manual time required, while still maintaining a high quality cut out. As such, the following summary discusses the merits of the various presented state-of-the-art systems with respect to the amount of user interaction, how effectively user information is exploited, the level of interactivity, the conditions when the system fails, and the quality of obtained cut outs.

5.5.1 Quality of Results

The best endorsement for the quality of spline based cut out techniques is their wide-spread use throughout the entire video post production industry. Currently, the large majority of (natural) video compositing tasks in professional settings use rotoscoping to extract objects. Due to the nature of the cut out strategy, rotoscoping provides its own quality assurance, where the desired cut out quality determined by the visual inspection of operator. That is, the number of splines placed around the object is determined by inspection. More complicated objects require more splines and consequently more time to manipulate them. Rotoscoping can essentially be viewed as one long corrective procedure; if the object is bounded closely by a curve, it is cut out. If not, move the curve such that the object is cut out, and repeat for all frames in the sequence. The quality of the results are therefore governed by the skill of the operator.

In pixel based video segmentation schemes, the quality of the segmentation is largely dependant on how user information is supplied and propagated through the video. The two systems that stand out in terms of quality are the Video Snap-cut and LIVEcut systems of Bai et al. and Price et al. respectively. Although the success of any semi-automatic extraction scheme depends on the level of user input, the Video Snap-cut and LIVEcut systems appear to find a good balance between the level of user interaction with desired results. Other systems, such as GeoS by Criminisi et al., tend to favour a more interactive approach, requiring the user to make many small adjustments that are reflected in the segmentation in near real-time. The resultant segmentation may look very good, but it can take a long time to achieve it. On the other hand, motion based segmentation schemes attempt to remove the user from the work flow, and also require certain constraints on the motion "layers" in the scene before becoming practical. The Video Snap-cut and LIVEcut systems exploit detailed user information and smart weighting of complementary feature cues. In practice, this means that high quality video cut outs are obtained quickly and intuitively, similar to the extraction of objects in still images.

in the approach by Levin et al., no starting point is needed.

5.5.2 User Interaction

It is clear that spline based methods currently require the greatest amount of user interaction to extract an object. With the exceptions of scenarios where semi-automated tracking may help, key-frames need to be placed at most 5 to 10 frames apart for reasonably uncomplicated scenes, and even closer together for others. Given an object with a detailed boundary, it would not be uncommon for roto curves to have a few hundred control points to be moved per key-frame. However, the supreme advantage is that the results of rotoscoping are predictable; it is easy to see where key-frames should be placed, and the effect they will have. For example, the user will usually select key-frames at important motion junctions, such as the turning points in a walk cycle. During simple transition conditions, a rule of thumb might also be to make a key-frame every n frames to keep control of the matte. How roto spline information is propagated over time is also very intuitive; the locations of the control points are simply interpolated between frames. If the user does not like how the interpolation has captured the object at a particular frame, a new key-frame can simply be placed at that frame and the control points manipulated. In this way, the user has full control over how the cut out information is propagated over time. It is not so simple in pixel based systems.

As pixel based systems are semi-automatic, the user generally inputs some information, waits for the response (which could be a long time), then observes and corrects the response if needed. As mentioned before, it is not clear how corrections in one image should affect images in another, and so corrections may be repeated ad nauseam until the desired results are achieved. For example, a difficult to segment region may be corrected in one frame, and may require similar corrections in *every* subsequent frame. However, there are two simple ways to reduce the amount of time spent interacting with the system. The first is obvious; make the system operate fast enough such that it becomes highly interactive, such as GeoS of Criminisi et al. It does not matter then if the user cannot predict what will happen as the effort required to correct (or re-correct) the segmentaion is sufficiently low. This is also seen in the Video Snap-cut system of Bai et al., where an updated cut out following simple user corrections requires only the recalculation of the cut out in a local window, making the system very responsive.

The second way to improve user interaction is to try to provide the user with an advance view of what is going to happen. For example, the "Interactive Video Cut Out" system of Wang et al. allows a clear display of the "hard" user strokes (regions with fixed α) applied to the video by visual, interactive manipulation of the 3D volume. Although the exact segmentation may vary, the user has a reasonable idea where, and how, it will vary. The re-weighting of feature cues in the LIVEcut system of Price et al. provides a less explicit but very useful method for predicting how the segmentation will proceed. For example, if the user manually marks a difficult area in one frame, it is likely that the same difficult region will not need to be fixed in later frames. Presenting "diagnostic" foreknowledge to the user is a powerful idea, illustrating where a given stroke, scribble or bounding box should be placed to have maximum effect throughout the video. Levin et al. demonstrate this in their matting algorithm, where the eigenvectors of their connectivity matrix L show well connected regions. This diagnostic approach to video segmentation is very useful, and is explored in the next chapter of this thesis.

Regarding the user interactivity of the presented Bayesian [35] and Closed-Form matting [92] systems, the difference in the amount of user interactivity required to get accurate results from the two systems is large. The Bayesian matting framework requires a good, accurate tri-map in order to obtain high quality results. If the results of the system are not satisfactory, the tri-map must be corrected. Often it is unclear how changes to the tri-map will affect the resultant matte, for example, how does the matte quality vary with the size of unknown area? This leads to a large number of corrections. In the Closed-Form matting system, the preferred method of user interaction is by supplying paint strokes, and is generally able to produce higher quality mattes with far less user interaction than the tri-map. Additionally, the amount of correction time is significantly less. However, both systems are computationally intense, requiring at least half a minute of processing time for results to be returned. This means that the amount of user interaction should be kept to an absolute minimum for any practical use in 2D still images.

5.5.3 System Failure

In pixel based systems, the user will generally mark the object in a frame that represents the object well throughout the sequence (the first key-frame). The user lets the system run, and awaits the labelling results for the current or subsequent frames. As each of the pixel based video segmentation schemes presented rely on colour for the data likelihoods, the number of key-frames required depends on how the modelled object and background colour distributions change over time, with the amount of effort required per key-frame determined by how well separated the distributions are in colour-space. Additional key-frames (scribbles etc.) are then placed in order to correct any errors in the segmentation, and the system re-segments the video. In this way, key-frames are only added *reactively*, with the user placed in a constant "corrective mode". This happens for two reasons; firstly, it is difficult to predict where the algorithm is going to fail, and secondly, depending on how the system is set up, corrections made to one frame may apply to just that frame, or may apply to other frames in the video, possibly to frames that have already been corrected. The lack of clarity between what the user wants and how the system can provide it is a problem that needs to be addressed before pixel based cut out systems will be adopted as serious post production tools.

A large problem not addressed by pixel based cut out systems is how temporal consistency can be enforced. For example, consider an object with an ambiguous vertical edge. In one frame the object cut out may delineate the object more along the left side of the edge. In the next frame, a subtle change in lighting could adjust the actual object colour distribution enough to cause the labelling of the current frame to delineate the object more to the right side of the edge. Assuming this edge remains ambiguous for the rest of the sequence, when the resultant labellings

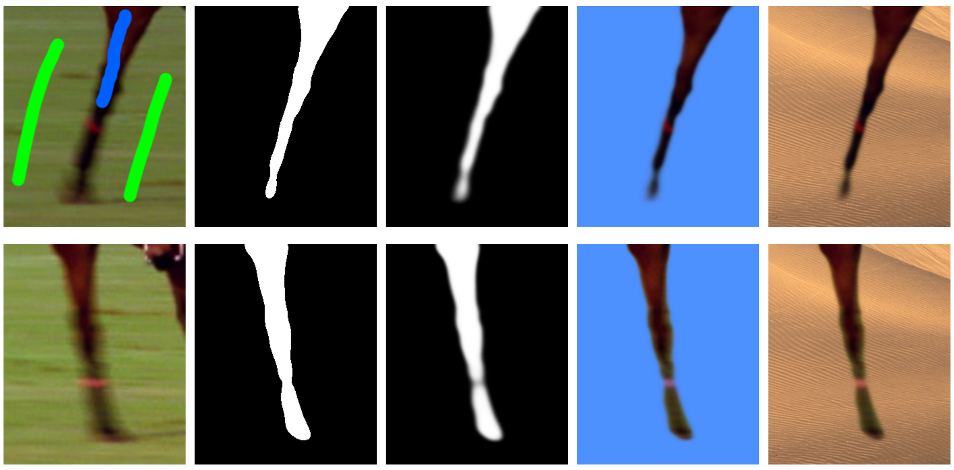


Figure 5.27: Example of temporal segmentation "popping". First column, two close-ups of the leg of a running horse taken 2 frames apart, exhibiting difficult, ambiguous object boundaries. The blue and green scribbles are used to mark the object and background in the first frame to model the associated colour distributions. Top row, the object is cut out (2nd image) and matted (3rd image) using the scribble colour distributions, and composited onto novel backgrounds (4th & 5th images). Bottom row, using the model colour distributions from the first frame, the object is cut out in this frame. Notice how small changes in the actual object and background colour distributions cause large differences in the resultant cut out. The popping effect in this contrived example is a function of both a slow camera shutter speed (causing excess motion blur), and colour quantisation artifacts induced in the video compression. In this case can be alleviated by improving the overall image capture, through higher frame capture rates and compression quality for example. However, the underlying problem is always that user information imparted in one frame may not be valid for later frames, and often manifests as this "popping" effect.

of the sequence are viewed as a video, the region at the edge will appear to "pop", as the label values rapidly switch around the edge. An example of this temporal cut out inconsistencies is shown in Figure 5.27. This problem is sometimes automatically corrected in the later matting stage, however most matting schemes do not correct for significant problems during the coarse cut out stage. As spline based methods generally ignore the image data, relying instead on the eye of the artist, this problem of temporally inconsistent mattes is less of a problem. Assuming the artist is able to notice the temporal artefact in the final matte, it can be remedied simply by moving the control points of the feather curve until the matte appears temporally consistent. In pixel based systems, an additional tool would need to be created to somehow force smoothness over time, however it is not obvious how this can be done. Wang & Cohen [179] propose solving the matting problem over several frames to encourage temporal stability, however it would more sensible to address this problem earlier in the object segmentation stage.

In relation to user control, all of the pixel based segmentation schemes (for still image or video) use prior information defining how the spatial neighbourhoods of pixels, and how they interact. They also use gradient information as a cue to help segmentation, i.e. neighbouring pixels do not have to share the same label if there is a high intensity gradient between them. Implicitly this encourages object labellings to be delineated by gradient boundaries. This works well when a tight cut out around the object is desired, however in post production there are often cases where the desired part of the frame (which may or may not contain an object) to be extracted is not well bounded. For rotoscoping, the cut out task is the same as before, but pixel based systems will fail, either because the "object" and background colour distributions can not be sufficiently separated or because no clear boundary exists. Given that modern pixel based segmentation schemes were not designed to handle such situations, this is one case where the objectives of the spline- and pixel based segmentation strategies differ.

5.5.4 On Updating

In many of the pixel based systems discussed, the main information used to segment the object in later frames has been the colour information derived from the scribbles. The "Distance-Cut" system of Bai & Sapiro used proximity to scribbles as an additional likelihood. When the system fails, and the user corrects the matte with additional corrections, it is unclear how these corrections should affect the rest of the video. For example, should it be assumed that everything up to this point is correct, and that from now on the original colour distributions (and scribble positions) should be discarded? Or should the changes be applied to future frames, and retroactively to previously segmented frames? In their "LIVEcut" system, Price at al. allow the effects of local user corrections to influence the segmentation of future frames, implicitly forcing the segmentation to proceed in one direction. However, a case can be made for either propagation strategy. Clearly, this is a factor that can affect the frequency of frames to be corrected (key-frames) in pixel based systems.

From the research in spline based object cut out (or the lack thereof), it appears that rotoscoping is viewed somewhat as an occupational trade; technological enhancements may improve productivity, but the quality of a matte and the time taken to obtain it will always determined by the skill of the operator. On the other hand, an objective of pixel based systems is to substantially lower the skill level needed to obtain high quality mattes by automatically cutting out the object in the entire video given a small amount of user information. All of the state-of-the-art systems previously discussed have demonstrated significant advancement in high quality, semi-automatic video segmentation. They have shown how small amounts of user interaction can successfully extract objects over long video sequences, and how recent optimisation strategies can be exploited to allow real time user interaction with the system. Clearly both pixel- and spline based systems each have their strengths and weaknesses.

5.5.5 Improving Object Cut-out with Local Image Features

In the following chapters, some of the issues with current cut out techniques are examined and used to drive improvements, through the use of local image features. In particular, the following topics are considered.

Choice of Feature Space All of the pixel based schemes use colour as a large (or only) part their data likelihood. It is posited in this thesis that colour can not be presumed to be temporally consistent for long periods of time. Feature points have been shown to match content across extremely temporally disparate frames, through either correspondence matching or object detection frameworks. In the following chapter, an object segmentation strategy is proposed that uses feature points to propagate information further through a video sequence than is typically possible with colour information.

Segmentation Diagnosis The ability to instruct the user where and when to place strokes, scribbles or key-frames is a powerful tool for high quality video segmentation. Considering that many semi-automatic segmentation schemes do not provide real-time, interactive cut out, the next best solution is to provide the user with some fore-knowledge as to how their input will affect the overall segmentation. Using the feature based object detection framework discussed in an earlier chapter, a cut out diagnosis tool is presented in the next chapter that clearly and concisely shows the user the optimum frames to mark. This dramatically reduces the amount of trial-and-error guess work of semi-automatic segmentation.

User Information In the presented pixel based video segmentation schemes, the user is asked to mark video frames by strokes. While it has been shown that reasonable results can be obtained from this minimal amount of information, this thesis shows that more concise definitions of what *is* and *is-not* object supplied by the user can be propagated more effectively through the video. The idea is that the quality of the user input in one frame is directly related to how well the video cut out system can automatically segment other frames. For the proposed feature based segmentation, the input to the system are not strokes or rectangles, but rather 2D mattes generated by the user by whatever means (i.e. Grab-Cut, Distance-Cut, GeoS, Photoshop etc.). This is the approach of both the LIVEcut and Video Snap-cut systems. These input mattes are a complete, explicit definition of what is the object and background. For example, if the user inputs detailed object mattes, the resultant cut outs throughout the sequence will be detailed. On the other hand, if a coarse cut out of the object is presented, the resulting video cut out will be coarse (and still contain the object). In this way, the user is free to define the level of detail required from the matte in the same way as a rotoscope Bezier curve.

Correction As discussed, the segmentation will fail as the actual object and background colour distributions deviate over time. Using local image features as the feature space means that the segmentation will fail if reliable feature matches cannot be found. The usual tool for correcting cut outs from video segmentations is to paint a corrective scribble over the region where the segmentation failed. A new user tool based on Bezier curves and a feature matching energy minimisation framework is presented in a later chapter to interactively establish feature correspondences in difficult regions. This new useful tool allows the single correction to be applied to many other *relevant* frames at the same time.

Chapter 6

Feature Based Object Segmentation[1]

Accurately segmenting objects in video is a difficult and time consuming process in modern post-production houses. Automatic systems may work for a small number of frames, but will typically fail over longer video shots. This work proposes a semi-automatic, feature-based system to perform object segmentation over longer sequences. An information-retrieval (IR) based diagnostic instructs the user to manually segment frames containing representative instances of the object. The user supplied masks are then propagated to the remaining un-segmented frames and used to bootstrap the automatic segmentation for these frames. The work presented in this chapter dramatically reduces the manual workload required to segment a video sequence, allowing longer and more accurate object mattes.

6.1 Introduction

In media production, video compositing workflows rely on the artist's ability to accurately select and extract objects from shots of video. Semi-automatic segmentation schemes aim to reduce the work-load of the artist by "auto-completing" the repetitive parts of the manual object cut out. This makes sense, as the object generally does not undergo dramatic transformations between consecutive frames. The idea is to leverage the information supplied by the user in one frame to automatically segment other frames, an example is shown in Figure 6.1. From the video segmentation review presented in Chapter 5, the factors that are most important for post production quality video segmentation are; *a*) exploiting user information to maximum effect, reducing the amount of overall user interaction *b*) giving the user a clear idea of "cause and

[1]Results from this chapter have been published as:

"Feature-Cut: Video Object Segmentation Through Local Feature Correspondences" by Dan Ring and Anil Kokaram, in *Proceedings of International Conference of Computer Vision (ICCV) Workshop on Video-oriented Object and Event Classification (VOEC)*, Kyoto, Japan, October 2009

"Information Retrieval Assisted Object Segmentation In Video" by Dan Ring and Anil Kokaram in *IET European Conference on Visual Media Production (CVMP)*, London, UK, October 2008.

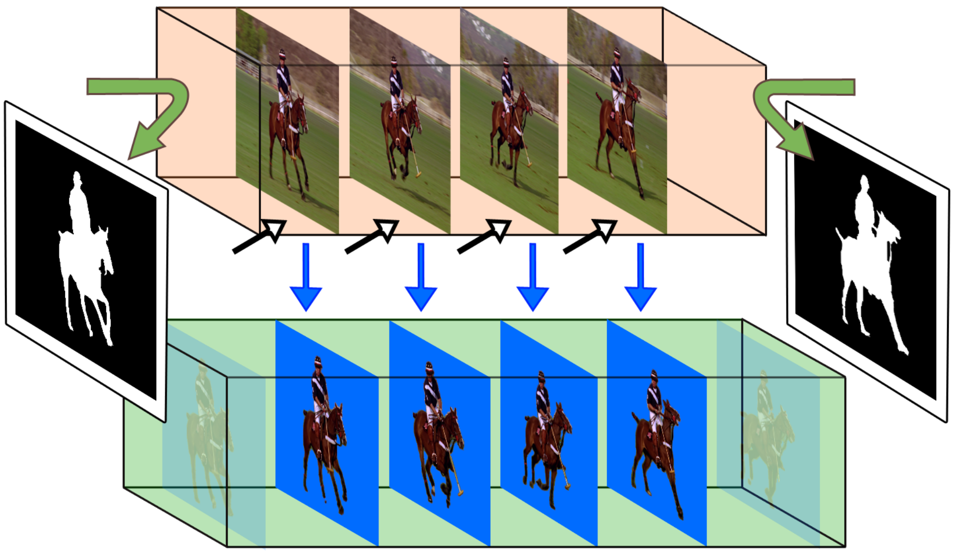


Figure 6.1: The objective of this work. The user supplies initial object mattes for a few frames, in this case 2 (black and white, far left and right). Information from these mattes is propagated using feature points, allowing the object to be quickly and accurately segmented in other frames automatically (bottom). This approach dramatically reduces the manual effort required to segment an object in video.

effect", i.e. if user information is supplied in one frame, its effect on later frames is predictable, and *c*) responsive, near real-time user interaction. The novelty of this work lies in exploiting feature-based object detection and localisation research to address these criteria, allowing the propagation of user-supplied segmentation information through video over longer sequences[2].

As discussed in Chapter 2, sparse local image features such as Harris-Laplace [115] and SIFT [105] are highly invariant to various photo-metric and geometric conditions, allowing them to be reliably re-detected throughout a video sequence. Various research has demonstrated the use of feature-points in video for tasks such as object retrieval [162], tracking [189], shot boundary detection [152] and more. The work in this chapter exploits the temporal stability of feature-points to allow accurate object segmentation over longer sequences of video. By establishing feature point correspondences between frames, it is possible to "pull" information from user supplied mattes into frames to be segmented. Unlike typical dense motion field matte propagation, feature correspondences can be identified between non-consecutive frames, allowing the propagation of user information from several temporally disparate frames at a time.

Similar to the previously discussed Video Snap-cut [11] and LIVEcut systems [136], the presented feature based cut out system takes manual cut out examples of the object as input,

[2]It should be noted that the presented segmentation system was developed in parallel, but is closely related, to the Video Snap-cut system of Bai et al. [11]. The differences and similarities will be highlighted in the discussion at the end of this chapter, in Section 6.5.

as shown in Figure 6.1 (far left, far right). The idea is that a complete labelling of the object contains a more precise definition of what the user regards as object or background, compared to scribbles for example (although scribbles or strokes can be used to create the initial manual segmentation). The supplied manual labellings are then used to segment subsequent frames. The novel aspect in this work is to propose that by using feature correspondences, user supplied information from mattes can be propagated further and more reliably than motion or colour features. In addition, a diagnostic technique is presented that uses feature adapted text-based IR techniques to indicate to the user the optimal "key-frames" to manually segment, in order to minimise the overall user interaction.

Section 6.2 details how IR techniques are applied to assisting the user with segmentation, Section 6.3 describes the propagation of mattes using feature correspondences, Section 6.4 gives video object segmentation results from this system, with a discussion presented in Section 6.5.

6.2 Information Retrieval Assisted User-Interaction

In any semi-automated system, there is always the question of where and when the user needs to interact with the system to achieve the greatest effect with minimum effort. For example, consider segmenting an object in a still image. For a reasonable segmentation, the user should not need to label every pixel manually. Instead, a few pixel samples *representative* of the object and not-object colours can be selected, and used to segment the rest of the image. In this example, the choice of representative object and not-object colour samples determines the quality of the segmentation results. In video object segmentation, the problem becomes more difficult. The transient nature of video means feature spaces such as colour or shape will change over time, representative selections of the object in one frame may not sufficiently describe the object in later frames.

One of the issues addressed in this chapter is how to select the most representative instances of the object throughout the video. That is, finding the instances of the object that when manually segmented, have the greatest effect when automatically segmenting later frames. The idea is that by initially identifying and labelling a "core" set of object instances, the overall effort of manual interaction is reduced.

The "Obj-Cut" algorithm of Kumar et al. [88] demonstrates the use of representative object "parts" in segmentation. This technique relies on extensive prior training of object parts, parts which must first be segmented themselves. Sivic et al. [160, 161] apply text based information retrieval (IR) techniques to feature points, allowing the fast indexing and retrieval of user-defined objects. The presented work extends the feature-based IR framework of Sivic et al. to detect frames in which the appearance of the object changes significantly, forming a set of representative instances of the object. By only hand-labelling the frames exhibiting a significant departure in appearance from the object model given from previous labellings, the amount of work needed to segment the entire shot is kept to a minimum.

The work in this chapter adapts this “bag-of-words” model (as presented in Chapter 2) to assist the user in selecting the optimal frames to manually segment. Recall how an image (or “document”) can be represented by tf-idf weighted vectors of feature occurrence frequencies, and that the similarity between two documents is given by the cosine difference between the pair of frequency vectors. For every video frame in the sequence, feature points are calculated and assigned to one set. Features belonging to the object are then identified from the manually extracted mattes, and are assigned to another set. Using the tf-idf weighting and cosine difference, the similarity between the vectors belonging to the collection of object features and the features for a given frame is measured. Given that the object should exist in every frame of the shot (for most object segmentation tasks), the document similarity measure indicates how reliability objects can be re-detected using the previously extracted matte information. Frames with low re-detection reliability indicate departures from the known object appearances, and are good candidates for the user to manually extract as key-frames. Manual segmentation of these frames will improve the overall segmentation accuracy more than manual segmentation of frames similar to those already manually segmented. Therefore, by using the adapted IR framework to select to manually segment, the effect of user interaction is maximised. The following section describes the set-up and use of the adapted IR framework.

6.2.1 IR System Specifics

The specific details of the user information diagnostic system are now presented.

Choice of Feature Detector The criteria for selection were limited by the practical constraints of the system; finding feature-points in video is a lengthy process, and as such, a detector scheme that balances the tradeoff between accuracy and speed is needed. For this reason, the feature point detector used in this work are Hessian-Laplace points, as outlined in [117]. The image data surrounding the detected features are then represented using SIFT descriptors [105].

Feature Codebook Generation The number of clusters N_c (codebook indices) are trained offline on a large number ($> 50,000$) of descriptors taken at random from the video to be segmented. These descriptors are from both the object and background in the image. In video segmentation scenarios, the scene content within the video clip to be segmented will probably not vary as dramatically as those in the feature film detection scenario of Sivic et al., or object recognition scenario of Nister et al. As such, the dictionary of possible descriptors can be a lot smaller than the 1M size of Nister et al., N_c is typically between $1,000 - 5,000$ depending on the complexity of the video. After clustering of the random descriptor set, feature descriptors from the video frames are quantised to the nearest cluster centroid, and represented by code-book values i corresponding to centroid number. An offline clustering system is presented here as it is assumed the entire video is going to be processed at once. However, a practical application would generally prefer an “on-line” system with less pre-processing. This can be achieved by

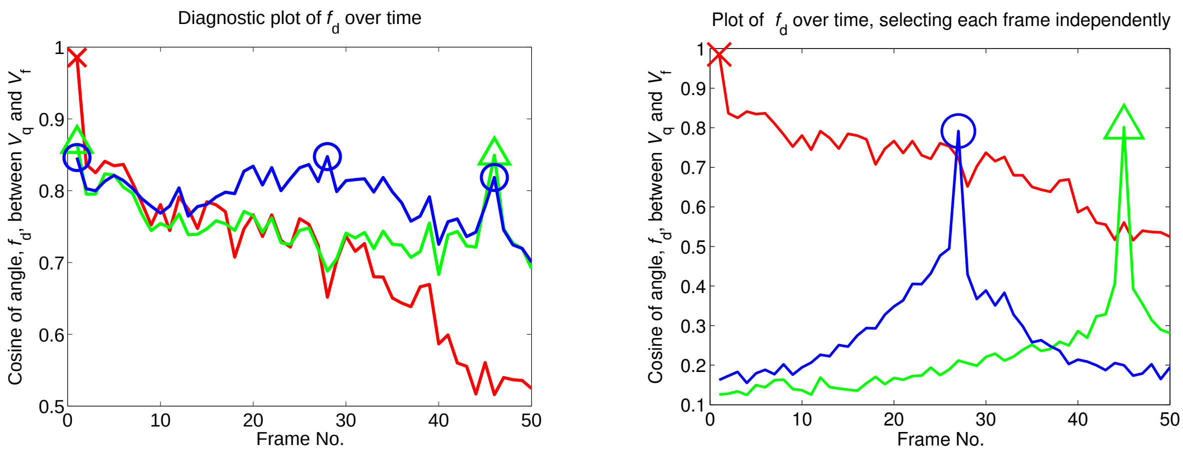


Figure 6.2: Example of object detection reliability over time after selecting instances of the object at various frames of the "quadbike" sequence. Left, the plots show the values of f_d following manual selection of the object in frames 1 only (red, cross), frames 1 and 46 (green, triangles), and frames 1, 28 and 46 (blue, circles). Notice that the detection reliability improves with each successive user frame added. By using this diagnostic plot to select the optimum frames in which to re-select the object, the detection performance f_d increases substantially throughout the sequence. Right, for comparison, the plots show the effect of selecting frames 1, 28 and 46 independently of each other. The sharp peaks of frames 28 (blue) and 46 (green), in contrast to the relatively high values of selecting frame 1 (red), highlight the fact that they contain distinct instances of the object, and should therefore be selected as representative frames. Note that the plots of f_d over time for the selection of frame 1 (red) are the same in the left and right figures, the vertical scale in the left figure was adjusted for clarity.

doing a one-time a-priori clustering on a large random set of descriptors from representative video footage[3].

6.2.2 Measuring Object Detection Reliability

From the manual frame labellings, a model of the object is given by the list of quantised feature indices taken from features found belonging to the interior regions of the object. As discussed in Chapter 2, the Vector Space Model [59] can be used in conjunction with *tf-idf* frequency re-weighting to measure how reliably the object can be detected in a frame f, given the feature information about the object. The idea is that if the object can not be reliably detected in frame f (even though the object is known to be present in the frame), then the system probably does not have sufficient feature information to segment the object accurately, and needs additional user input. The detection reliability measure proceeds as follows.

[3]This only needs to be performed once if the random set describes enough of the possible feature descriptor space using a sufficient number of clusters. The success of PCA-SIFT [81], and the strong relationship between PCA and k-means clustering [45] indicate that this is achievable

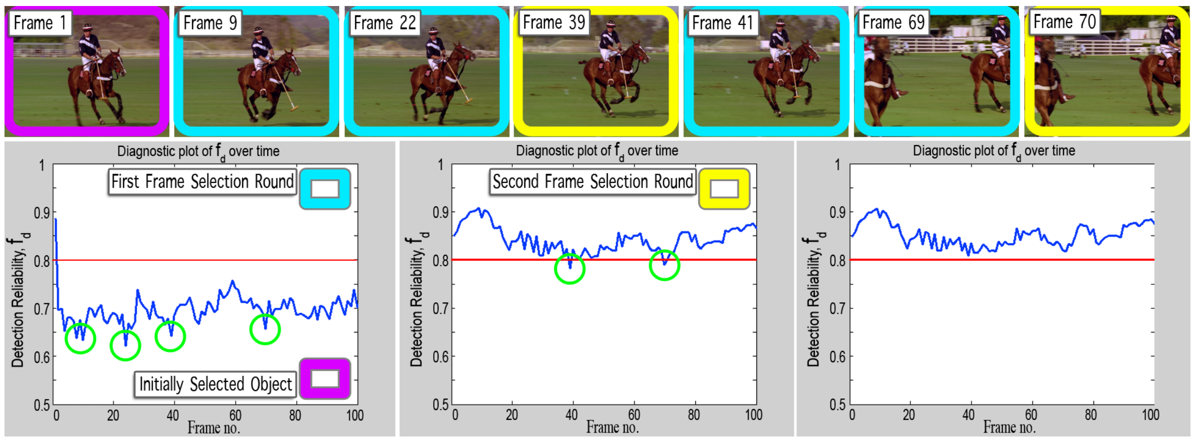


Figure 6.3: Example of using the detection reliability to select the optimum frames to be manually segmented. Given an initial object selection in frame 1 (top row, magenta), the system calculates the detection reliability, f_d, throughout the sequence (blue, bottom row, left). In this example, the quasi-periodic appearance of f_d is related to the gallop cycle of the horse. For this example, the objective is to select enough frames to raise f_d above a threshold of 0.8 (red lines, bottom row). Although the majority of f_d values are below 0.8, it is expected that selecting only a few frames will improve overall detection. Hence, frames 9, 22, 41 and 69 (cyan, top row) are selected first, chosen by the minima of f_d in the first plot (green circles). Note that these frames are reasonably different in appearance from frame 1, and from each other, and correspond to representative frames of the horse's gallop cycle (top row). The manual selection of the four frames is performed, and the detection reliability is calculated again for the updated object model (centre, bottom row). This time only two additional frames are flagged as requiring selection. For the last time, the detection reliability is calculated (right, bottom row), and it can be seen that all f_d values are now above 0.8, indicating that the object instances manually selected so far represent the object well throughout the entire sequence.

Features belonging to the object from 1 or more manually segmented frames, and are represented by an artificial "frame" q. The tf-idf weights for frames f and q are calculated (i.e. $t_{i,f}$, $t_{i,q}$, $\forall i \in \{1, \ldots, N_c\}$), and frames f and q can now be represented by vectors of their weighted feature occurrence frequencies, $\mathbf{v}_f = (t_{0,f}, t_{1,f}, ...t_{N_c,f})^T$ and $\mathbf{v}_q = (t_{0,q}, t_{1,q}, ...t_{N_c,q})^T$ respectively. The frame f is then "queried" for the object q by calculating the normalised scalar product (cosine difference), f_d, between the two vectors $\mathbf{v}_f$ and $\mathbf{v}_q$,

$$f_d = \frac{\mathbf{v}_q.\mathbf{v}_f}{\|\mathbf{v}_q\|\|\mathbf{v}_f\|} \ . \tag{6.1}$$

How well the selected object can be detected in the current frame is given by the angle cosine f_d. A high cosine difference indicates many of the object features are present in the frame f, and means the object can probably be reliably re-detected in the frame. The value of f_d will vary depending on the number of features present, and the number of additional user feature selections. An example of this is shown in Figure 6.2. Fig. 6.2 (red,cross) shows the values of

f_d are observed over the video for a single manual object selection at frame 1. As the object changes appearance, features selected in frame 1 disappear, making the object more difficult to detect in later frames, causing f_d to decrease over time. More manual selections of the object are needed to improve detection performance. In the case of the initial selection (red line), the point at which the object is least reliably detected is at frame 46, so this frame becomes the next to be manually segmented. Following the object re-selection in frame 46, a sharp improvement in detection reliability is seen in Figure 6.2 (green, triangle). Again, the frame of minimum detection reliability, frame 28, is chosen to be re-selected. The detection reliability is again improved following the manual object selection of frame 28, shown in Figure 6.2 (blue, circles). By using the plot f_d, the user can iteratively diagnose when the object is less likely to be reliably re-detected, and find the optimum frames in which to re-select the object to improve detection. If the values of f_d for the sequence are sufficiently high (this threshold depends on the video and the desired accuracy of the user), the object is considered to be reliably detected, and the features of the "object frame" q are assumed to be representative of all apparent instances the object in the video sequence. A more involved example is shown in Figure 6.3. The manually selected frames are now used to propagate the user mattes throughout the video.

6.3 Feature-Based Matte Propagation

Imagine two frames from a video, one to be segmented and the other already manually segmented, called the target and source frames respectively. The goal of this work is to segment the target frame by "pushing" matte information from the source frame using correspondences between feature points. Unlike so-called "dense", per-pixel motion vectors, each pixel in the target frame to be segmented is not guaranteed to have a corresponding pixel in the source frame. Feature correspondences between target frame and the source frame are only established between a pair of image regions if the regions are sufficiently similar. This means that if a pair of regions in the target and source frames are too dissimilar, no correspondences will be identified there, and it becomes more difficult to label the region pixels as object or not-object. However, if a correspondence between two regions can be found, the target pixels can use the matte information in the corresponding source region with certainty, greatly increasing the power of inference in the region. It is clear that the segmentation accuracy will be largely determined by the number and quality of the feature matches.

6.3.1 Identifying Feature Correspondences

A video to be segmented is composed of the set of frames in which the object has already been manually segmented (the source matte frames S), and the set of frames to be automatically segmented (target frames T). Given a frame from the target set, the task is now to find feature matches between the target frame and each of the frames in the source frame set S. There are

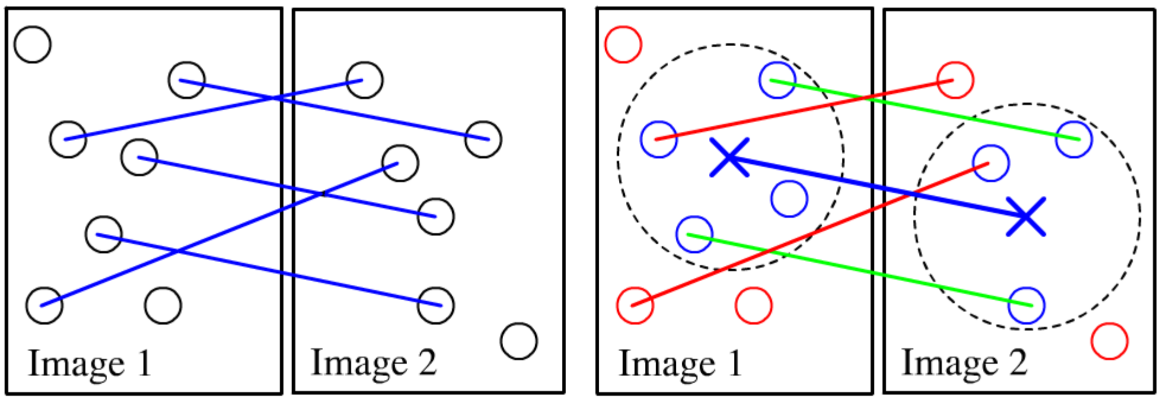


Figure 6.4: Example of feature matching by requiring at least k similar matches within a radius. Left, matches (blue) between features (circles) in two images are initially found using the "nearest-neighbour" descriptor distances. Right, a candidate match (blue line) between two features (blue crosses) is considered. Inside the radius (black dashed line), two similar matches can be seen (blue circles, green lines). In this example, the candidate match would be rejected if the threshold k were greater than 2. This match evaluation continues for all the candidate matches in the left image. A practical example of this spatial consistency constraint is shown in Figure 6.5.

many techniques for finding matches between sets of features, as discussed in Chapter 2. The simplest form of matching is the "nearest-neighbour" technique, where each feature point from one image is matched to the feature in the second image with the lowest descriptor distance, and if the distance is below a certain threshold. This matching strategy does not take the actual feature locations or surroundings into account, operating only on image patch appearance given by the descriptor distances. Spatial consistency constraints are included to help reduce the number of incorrect matches due to descriptor-only matching. To avoid the computationally expensive, or model specific spatial consistency schemes of Torresani et al. [173] and [172] respectively, the simpler spatial consistency check of Sivic et. al. is used, requiring that at least k similar feature correspondences lie in the vicinity of a potential match [161]. A diagram and example are shown in Figures 6.4 & 6.5. This is both fast and reliable at removing typical bad matches.

6.3.2 Speeding-Up Nearest Neighbour Descriptor Matching

For video segmentation schemes to be useful, they should be reasonably quick to produce results. Feature detection of course introduces its own computation overhead, however matching features is the most computationally intensive task in the system. As discussed in Chapter 2, there are several ways of reducing the time required for nearest neighbour feature matching. In this work, redundancy in the detected descriptors is exploited via PCA analysis. The idea now is to speed-up matching by transforming the descriptors into a space that will reject a match during

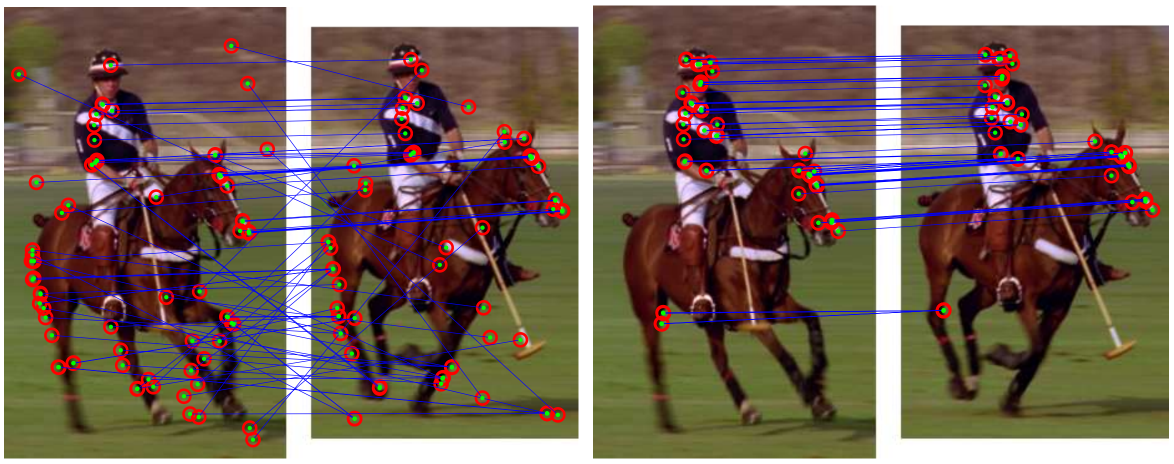


Figure 6.5: The advantage of including spatial consistency checks; "Nearest-Neighbour" matching (left pair), and requiring at least k similar matches within a radius (right pair). Although the number of feature matches is lower when the spatial consistency constraint is applied (right), the number of incorrect matches has dramatically decreased compared to the right image.

a descriptor pair comparison as soon as the descriptor distance calculation exceeds the current minimum distance or a pre-defined threshold. This was covered briefly in Chapter 2, but is now discussed in greater detail.

Similar to the k-means clustering in Section 6.2.1, a large set (>50,000) of descriptors are taken randomly from throughout the video sequence. The covariance $\mathbf{\Sigma}$ of the set of feature descriptors is decomposed by SVD, $\mathbf{\Sigma} = \mathbf{USV^T}$, to give the the eigenvalues $\mathbf{S}$ and eigenvectors $\mathbf{V^T}$. The entries along the primary diagonal of the eigenvalue matrix $\mathbf{S}$ correspond to the amount of variance that is contributed by each column of the eigenvector matrix $\mathbf{V^T}$. The elements of $\mathbf{S}$ and the columns of $\mathbf{V^T}$ are sorted by descending eigenvalues, i.e. the resulting diagonal entries of $\mathbf{S}$ become increasingly smaller. Imagine now a $[N \times 128]$ matrix $\mathbf{D}$ representing the set of N descriptors from an image. Using the eigenvector matrix $\mathbf{V^T}$ as a transform operation, the columns of the resultant $[N \times 128]$ descriptor matrix, $\mathbf{DV^T}$, are now ordered by decreasing variance per column. For example, values typically found in first column of $\mathbf{DV^T}$ will vary the most (having the highest variance), values in the the second column will have the second highest variance, and so on. This means that for a pair of descriptors, the distance between the pair increases the most over the first few descriptor element comparisons. Therefore, during a nearest-neighbour search between two transformed descriptor sets $\mathbf{D_1V^T}$ and $\mathbf{D_2V^T}$ from two images, candidate matches are likely to be rejected sooner into the pair-wise comparison, either because the running computed distance has exceed the current minimum (nearest-neighbour) match distance or it has exceeded a threshold, d_t. This results in significantly faster nearest-neighbour searching. Values of d_t used are discussed later in Section 6.3.5.

6.3.3 Propagating Mattes

The frames of a video to be segmented are divided into two sets, the set of source matte frames S, and the set of frames to be segmented T. The goal is to use the matte information from the frames in S to segment each frame in T. For each frame $t \in T$, matte information from S is then "pushed" along feature correspondences between the frame t and the frames in S. The questions now are what and how the information should be propagated?

As each correspondence is identified using feature descriptors, it makes sense that only matte information in the region around the matched features in the source and target frames should be considered. As SIFT descriptors are being used, the radius of the circular pixel region contributing to the descriptor is given by a multiple of the canonical scale of the feature point (typically 3 times). The work of Marszalek et al. [110] uses the idea of "shape masks" to detect and localise objects within a candidate image. This involves matching features between images in a database and the image in which an object is to be detected and localised. For each feature match between an image pair, pixels surrounding the features in both images are compared. If pixels between the image pair are deemed sufficiently similar, their locations are used to "vote" for the location of the object in the candidate image. A well-localised object will have many votes and appear as a well-defined object "mask". This work extends the idea of shape masks for localisation to the task of accurate object segmentation as follows, an example illustrating the idea is shown in Figure 6.6.

The fact that a correspondence has been made between a pair of images indicates that the image content between the images at the sites of the matched feature points is similar. As shown by the work of Marszalek et al. [110], the matte region around the feature in frame $s \in S$ can be "transplanted" onto the corresponding region in the to-be-segmented frame, $t \in T$. The actual "transplant" operation between two image patches is defined by the scale $\mathbf{D}$ and rotation $\mathbf{R}$ transforms between the pair of matched features. The matrices $\mathbf{D}$ and $\mathbf{R}$ are simply the change in the canonical feature scales and rotations:

$$\mathbf{D} = \begin{bmatrix} \left(\frac{\rho_p}{\rho_q}\right) & 0 \\ 0 & \left(\frac{\rho_p}{\rho_q}\right) \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \cos(\theta_q - \theta_p) & \sin(\theta_q - \theta_p) \\ -\sin(\theta_q - \theta_p) & \cos(\theta_q - \theta_p) \end{bmatrix}$$

where ρ_p and ρ_q, θ_p and θ_q, are the canonical scales and angles of the feature points p and q respectively. The matrix $\mathbf{u}$ defines the relative sites in the image patch, $\mathbf{u} \in [-r, \ldots, r] \times [-r, \ldots, r]$, where r is the radius or *extent* of the patch being transplanted, and is set to the radius over which the SIFT descriptor is calculated, typically $3\rho_q$. Given a source matte $M(\mathbf{x})$ from frame $s \in S$, the transformed matte using a single feature match (p, q) in the image patch

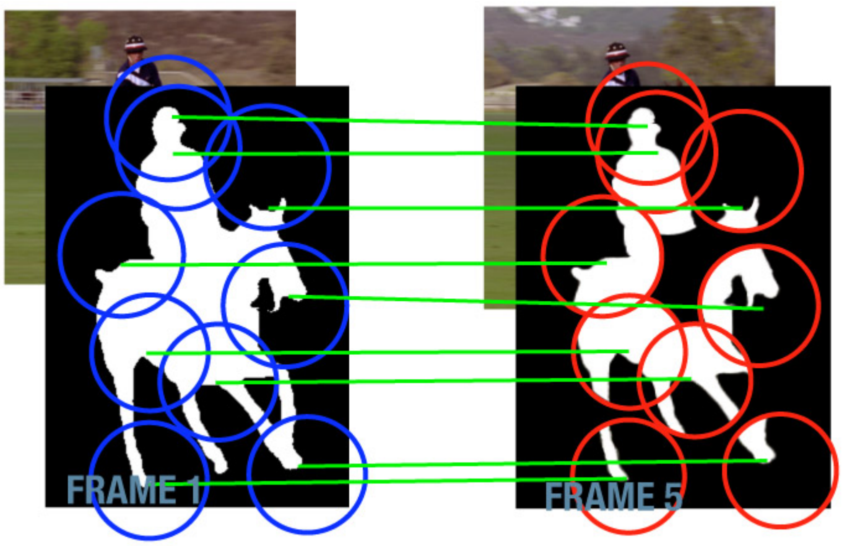


Figure 6.6: Example of propagating manually segmented matte information from frame 1 to frame 5 of the "Polo" sequence. Using feature correspondences between the frames (green), pieces of the user matte are pushed into frame 5. Each of the blue & red circular pieces represents the image region of the SIFT feature descriptor used to identify the match. Notice that the partial matte in frame 5 is missing information, and does not entirely delineate the object. However, there is still enough useful information in the partial matte to allow a more accurate object cut out be refined.

$\mathbf{u}$ is defined as:

$$M'(\mathbf{q}+\mathbf{u}) = M(\mathbf{p}+\mathbf{DRu}) \ .$$

Given that the image content between the feature pair (p, q) is already similar, the resultant transplanted matte should appear reasonable in the context of the new image, yet is probably not perfect. To help correct potentially incorrect matte pixels, the warped region matte $M'(\mathbf{q}, \mathbf{u})$ is weighted by the pixel similarity of the warped matched pixel regions[4] , $\Delta I = 1 - \|I_t(\mathbf{q}+\mathbf{u}) - I_s(\mathbf{p}+\mathbf{DRu})\|$. This is the propagation of matte information for a single feature match.

Consider now, extending this idea to include all the feature matches (p, q) between s and t, call this set $N_{(s,t)}$, for all the source frames $s \in S$. As in [110], $M'_f(\mathbf{q})$ and $M'_b(\mathbf{q})$ are the summations of the transplanted object and not-object partial mattes over all matches, defined as:

$$\begin{aligned} M'_f(\mathbf{q}+\mathbf{u}) &= \sum_{s\in S} \sum_{(p,q)\in N_{(s,t)}} g(\mathbf{u}, r) M_s(\mathbf{p}+\mathbf{DRu})\Delta I \\ M'_b(\mathbf{q}+\mathbf{u}) &= \sum_{s\in S} \sum_{(p,q)\in N_{(s,t)}} g(\mathbf{u}, r)(1 - M_s(\mathbf{p}+\mathbf{DRu}))\Delta I \ . \end{aligned} \quad (6.2)$$

$M'_f(\mathbf{q})$ and $M'_b(\mathbf{q})$ can be thought of as a "vote-space", where the values of M'_f and M'_b at a pixel site $\mathbf{x}$ are the number of features that "voted" for the pixel $\mathbf{x}$ to be object or not-object

[4]For simplicity, it is assumed here that the maximum value of $\|I_t(\mathbf{q}+\mathbf{u}) - I_s(\mathbf{p}+\mathbf{DRu})\|$ in this case is 1. However, for most practical cases this will need to be changed to suit the actual values of I.

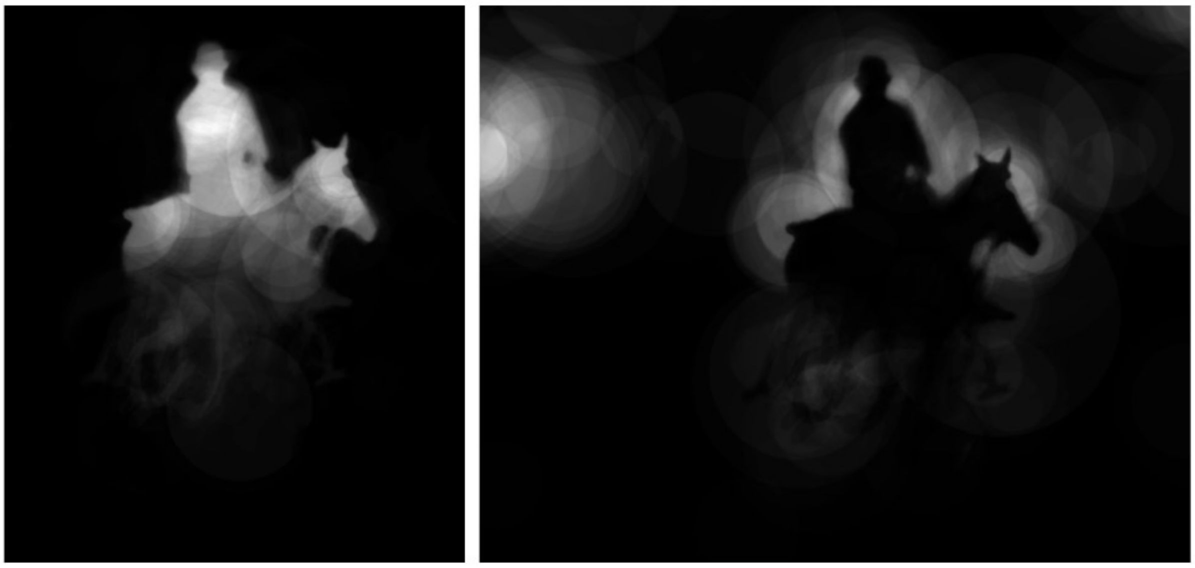


Figure 6.7: Example of propagated matte information from three matte source frames. The accumulated partial mattes of the object $M'_f(\mathbf{q})$ and $M'_b(\mathbf{q})$ background are shown on the left and right respectively. Brighter regions are the result of more feature matches in the region. Although there is relatively less information in the horses legs, there is still enough to create an accurate matte.

respectively. g is a circular window function of radius r, where $g(\mathbf{u}, r)$ is 1 if $\|\mathbf{u}\| \leq r$ and 0 otherwise.

As the source matte M_s is a binary matrix, the accumulation of pixels labelled not-object (a matte value of 0), would always equal 0. Therefore, object and not-object matte information needs to be propagated seperately, resulting in $M'_f(\mathbf{q})$ and $M'_b(\mathbf{q})$. An example of both accumulated partial mattes is shown in Figure 6.7.

The quality of the propagated mattes relies on the number and quality of the feature correspondences. For example, the "horse and rider" object mattes in Figure 6.7 were propagated using matches similar to those in Figure 6.5 (right). Notice that the horse's legs have relatively fewer matches than those for the horse's body and rider, and because of this, the mattes in Figure 6.7 are not as well defined for the legs. Thresholding the partial mattes as-is would not result in a useful matte. However, there is enough information provided by the partial mattes to extract a more accurate matte.

6.3.4 Refining the Partial Matte

Although the propagated mattes do a reasonable job of delineating the object, as seen in Figure 6.7, the mattes are far from useable. They can be used to boot-strap a more accurate segmentation process. Still image segmentation frameworks such as Grab-Cut [148] or Distance-Cut [9] achieve accurate still frame cut outs with relatively little user input. These algorithms model the colour distributions of user defined foreground and background regions to use as the object / not-object likelihood functions. In the presented work, the propagated mattes often contain more useful information than the derived colour information. It makes sense to use the propagated foreground and background partial mattes directly as a likelihood function. This work proposes to incorporate the partial mattes into the energy minimisation framework of Boykov & Jolly [25] to accurately segment the current frame. However, there is one problem that needs to be addressed.

Unlike the colour likelihood functions from GMM's [148] or KDE's [9], there are often regions in the partial mattes that have little or no foreground or background information. For example, relatively few matches were found on the horse's leg in Figure 6.7, which means there is simply no information to say whether the pixels in the region belong to the object or not. It is difficult if not impossible to segment these pixels on their own, however, it is expected that the object / not-object labelling should be "smooth". That is, label information from nearby pixels can be taken into consideration to help label the difficult regions. In the case of the horse's leg, partial matte information exists for the horse's body. As the horse's body and leg are a contiguous region, it is sensible that the horse's leg should be have the same label as the body. This implies a smoothness constraint on the label field.

Consider the frame to the segmented, I, indexed by pixel sites $\{\mathbf{x_1}, \ldots, \ldots, \mathbf{x_N}\}$, where $\alpha \in \{0, 1\}$ is the label vector assigning the pixels as background or object for α_n values of 0 and 1 respectively. The set of partial mattes for the frame is $M' = \{M'_f,\ M'_b\}$. Label smoothness is enforced by a Gibbs energy function [62]. The modified energy function and proposed data and spatial energies are defined as follows:

$$\mathbf{E}(\alpha, M', I) = U(\alpha, M', I) + \lambda V(\alpha, I, \mathbf{C}) \tag{6.3}$$

where:

$$\begin{aligned} U(\alpha, M', I) = & \sum_{n=\{1,\ldots,N\}} \exp -\gamma\big(M'_f(\mathbf{x}_n)\big)\alpha_n \\ + & \sum_{n=\{1,\ldots,N\}} \exp -\gamma\big(M'_b(\mathbf{x}_n)\big)(1-\alpha_n) \\ V(\alpha, I, \mathbf{C}) = & \sum_{(n,m)\in\mathbf{C}} \|\mathbf{x}_n - \mathbf{x}_m\|^{-1}\big(|\alpha_n - \alpha_m|\big)w_{n,m}\ . \end{aligned}$$

U and V are the data and spatial energy terms, with λ being used to weight the contribution between the two. $\mathbf{C}$ is the set of pair-wise neighbourhood cliques between adjacent "compass-point" pixels. The function $w_{n,m}$ is used to weight the spatial energy term of a pair of pixels by the difference in their values, defined in this work as $w_{n,m} = \exp\big(-\beta|I(\mathbf{x}_n) - I(\mathbf{x}_m)|\big)$. Intuitively, the spatial energy term V penalises pixels having different labels than their neighbours. If the scalar weight β is 0, V becomes the "Ising" prior, otherwise the neighbourhood penalty is weighted by how different the neighbouring pixel values are from the current pixel. This encourages smoothness of pixel labels except across edges. The scalar variable γ is used to weight the contribution of the partial mattes. From [25], the variables β and γ are defined as:

$$\begin{aligned}\beta = & \quad \left(2 \cdot E\left[\left(I(\mathbf{x}_n) - I(\mathbf{x}_m)\right)^2\right]\right)^{-1} \\ \gamma = & \qquad \left(2 \cdot E\left[M'(\mathbf{x})^2\right]\right)^{-1}\end{aligned}$$

where $E[]$ is the expectation operator, in this case over the whole image. By calculating β and γ in this way, β is adjusted accordingly for high and low contrast images, and γ is adjusted to accommodate the effects on the partial mattes from high and low numbers of feature points in the image. With the data and spatial energy terms introduced, the labelling of the current frame, $\hat{\alpha}$, is achieved by minimising the energy function,

$$\hat{\alpha} = \arg\min_{\alpha} \mathbf{E}(\alpha, M', I) \ .$$

The energy function is minimised using Graph-Cuts [87, 28]. The presented system has many variables that can be adjusted by the user to suit their needs. A selection of the parameters most critical to performance is discussed.

6.3.5 Parameter Selection

The system can be divided into two main stages; information-retrieval diagnosis and matte propagation, detailed in Sections 6.2 & 6.3 respectively. Assuming that the parameters for the feature detector are fixed, the choice of settings for the IR assisted user-interaction are limited to the number of descriptor space clusters to use, N_c. Once this number is sufficiently large enough to capture the variety of image structures present, typically $> 50,000$, the effect of N_c on the the detection reliability measure, f_d is minimal. The majority of important parameters belong to two parts within the matte propagation stage; feature correspondence matching, and matte refinement, shown in Sections 6.3.1 & 6.3.4.

6.3.5.1 Feature Matching Parameters

The performance of the presented system greatly depends on being able to identify correct matches between pairs of features. The two parameters that influence what candidate matches are selected are; the match threshold in PCA transformed descriptor space d_t, and the number of similar matches in a region k. Although the spatial consistency constraints will remove many spurious matches, the majority of bad matches will be removed by thresholding. As the SIFT descriptors are normalised such that the L^2 norm of each descriptor is 1, the range of distance values between a pair of descriptors is $[0, \sqrt{2}]$. The threshold d_t should strike a balance between Type 1 and 2 match error; if d_t is too low many matches will be incorrectly rejected, if d_t is too high, many matches will be incorrectly accepted. This system can handle missing data (through neighbourhood MRF information during matte refinement) better than it can handle

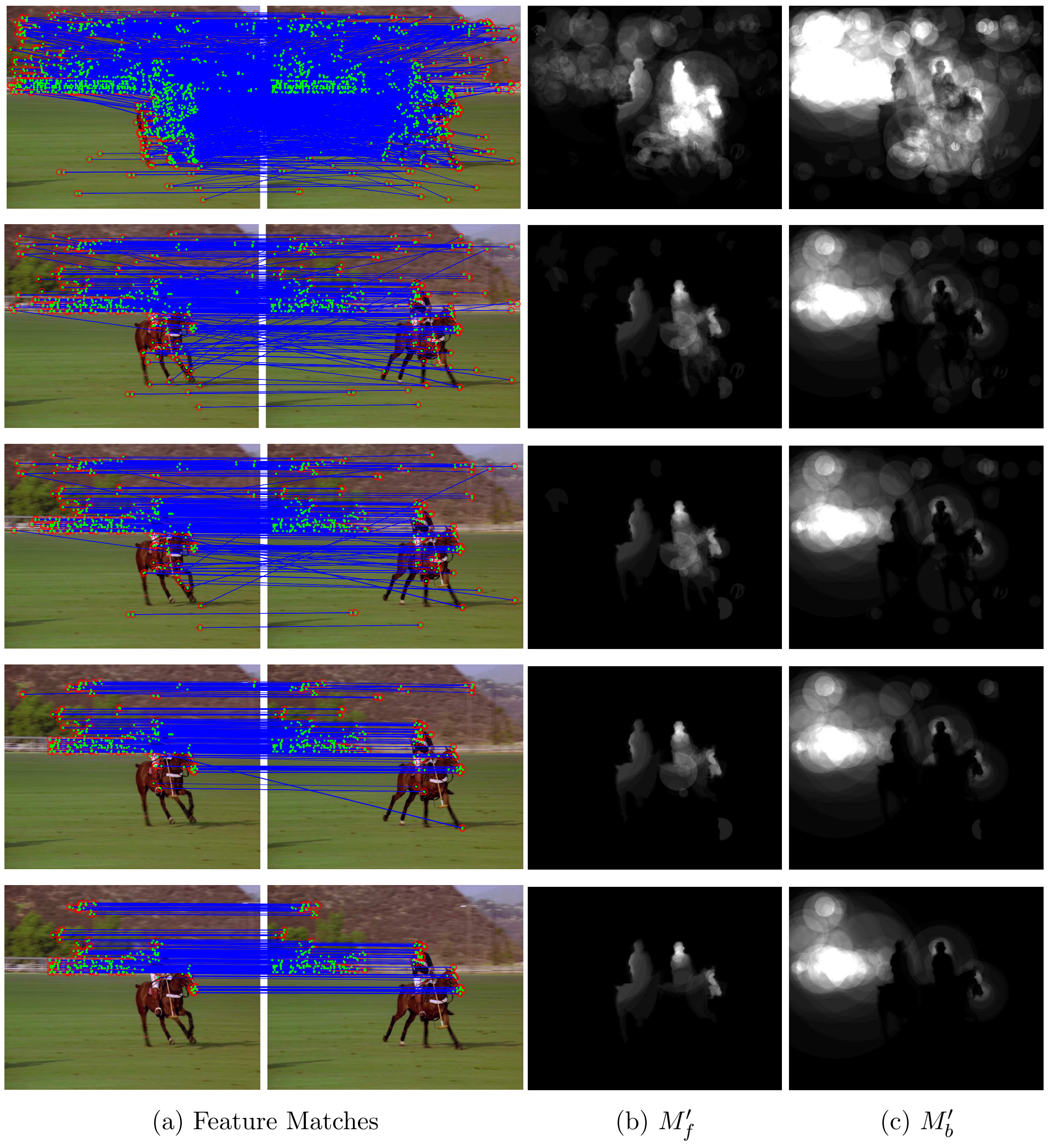


Figure 6.8: Results of varying the number, k, of similar nearby matches required to classify a match, against foreground and background propagated mattes. From top to bottom, $k = 0$ (no spatial consistency applied), 1, 2, 5 and 10. Without any consistency constraints the large proportion of incorrect matches (top-left) induce large errors in the likelihoods (top-right). Notice that the ratio of correct to incorrect matches increases with k, although the number of overall matches decreases. This lowers the amount of useful data in the likelihoods, leaving some areas without any information, for example the legs of the horse. The value for k needs to balance the percentage of correct matches versus the overall number of matches. For this "Polo" sequence, a value of $k = 2$ is chosen.

matte information propagated from spurious matches. For this reason, d_t is set conservatively low at 0.4.

The number of nearby matches k is related to the radius over which matches are considered. This radius should be based on how much a feature match should influence or be influenced by it's neighbours. Clearly, a radius of 100 pixels is too high, and a value of 5 is too low. For this work, a radius of 30 pixels captures the feature neighbourhood well for standard definition (720 by 576 pixels) resolution video, as used in the results later. The threshold of nearest matches k defines how strongly matches within the radius should agree with each other. If it is too low, then it is possible that false matches that are near each other by chance will be accepted. If k is too high, then regions with lower densities of features might not have enough correct matches. Examples of the propagated masks (object likelihood model) for various values of k are shown in Figure 6.8. For the presented results, the value of k was fixed at $k = 2$ for the "Polo" sequence due to relatively low numbers of features, and $k = 3$ for all others.

6.3.5.2 Matte Refinement Parameters

Once the likelihood energies, U, of Equation 6.3 have been calculated, the last remaining parameter to influence performance is the data to spatial energy ratio, λ. This parameter determines the amount of influence pixels have upon their neighbours. This is vitally important in the proposed segmentation system, as it allows for data in regions of strong likelihood to affect regions with weaker or no likelihood energies, for example, regions where no feature matches have been found. If λ is too low, the neighbourhood influence will be negligible and without a clear likelihood, the labelling will appear noisy. If λ is too high, the labelling will be over-smoothed. Clearly, for the trivial case of $\lambda = 0$ the problem becomes a standard MAP problem, and the labelling of pixels will depend on U alone, i.e. $\hat{\alpha} = \arg\min_{\alpha} U(\alpha, M', I)$. Although the choice of λ is dependent of the images supplied, there is a range of sensible values for λ that fit most image sequences quite well, examples of which are shown in Figure 6.9. For the presented results, a value of $\lambda = 0.05$ was used for the "Polo" sequences, as the likelihood energies for pixels belonging to the horse's legs are not strong, and a high λ value risks cutting them off. For the other sequences a value of $\lambda = 0.1$ was used.

6.3.6 User Interaction

To summarise the user interaction involved in segmenting a video, a typical interaction with the system is described as follows.

1. ***Select desired object in one frame.*** To give the system an initial idea of what the object to be segmented is, the user manually segments the object in a frame from anywhere in the video. The manual matte and feature points encapsulated by the segmented object are added to the sets of object mattes and features. This can be seen by the yellow arrows in Figure 6.10.

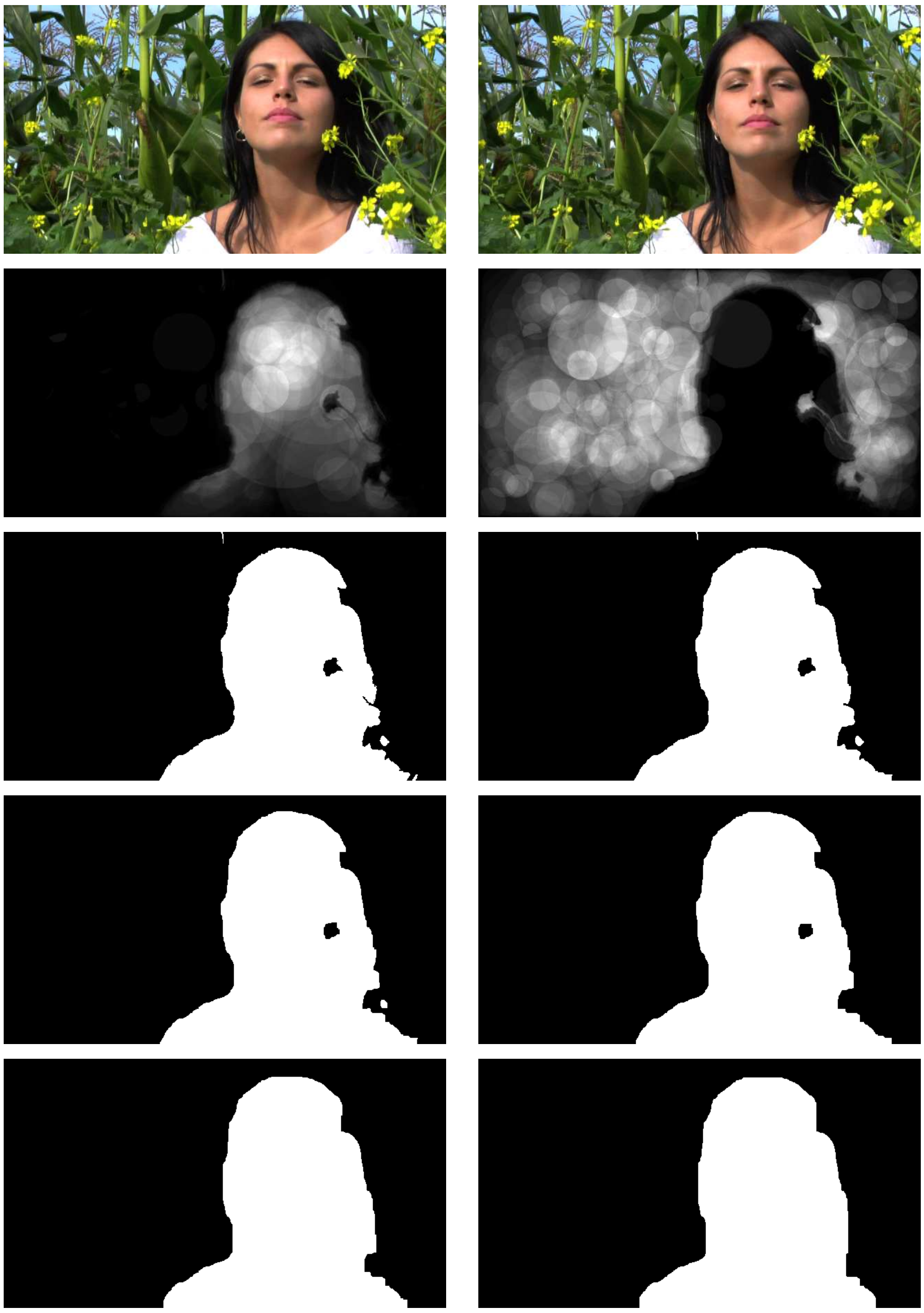


Figure 6.9: Effects of varying the data- to spatial-energy ratio parameter λ against segmentation results. Original sequence images, 5 frames apart, are shown in the top row. Object and Not-Object likelihoods are shown in the second row. Labelling results for $\lambda = 0$ (no neighbourhood smoothness), 0.1, 0.5, 1, 5, 10 are shown from the third row, left image to the bottom row, right image. Notice that for λ values of 0.1 and 0.5, the labelling is smooth while still preserving details such as the flower on her face. As λ is increased beyond 0.5, the labelling becomes over smoothed.

2. ***Analyse diagnosis plot.*** The object detection reliability is measured by comparing the set of object features to the features from all other frames. The user identifies frames the system will likely have trouble with, and manually segments the object in these frames, adding both the mattes and features to the object matte and feature sets. This process repeats until the user is satisfied with the overall level of detection reliability. This is shown by the red arrows in Figure 6.10.

3. ***Semi-automatic segmentation.*** Each of the remaining frames are then automatically segmented by the propagation of object mattes and subsequent refinement. If the resulting segmentation for a frame is not satisfactory, the user can manually touch-up the matte. The updates to the manually touched-up frames and features are then added to object sets, ensuring that any information supplied by the user is collected and incorporated into the system. The semi-automatic process can be seen by the blue arrows in Figure 6.10.

6.4 Results

To compare objectively the results from the presented algorithm, ground truth mattes were created for four video sequences using the "Quick-Select" tool in Adobe Photoshop [1]. Examples of the ground truth are presented in Figure 6.22. The sequences chosen exhibit various photometric conditions that often make segmentation difficult, such as highly-textured scenery, motion blur, flat texture-less regions, and low colour contrast. For the purposes of these results, the automatically generated object mattes were not manually touched-up afterwards. The number of Hessian-Laplace features calculated per image was capped at 2000 per frame. Results and details of the sequences are presented in Table 6.1. The processing times given are from a 1.8Mhz Intel Dual Core2 processor.

Table 6.1: Segmentation results compared to ground truth. $\bar{t}$ is the mean time for the system to automatically label a frame, $\bar{t}_m$ is the mean time for manually labelling a frame, and $\bar{\epsilon}$ is the mean error per pixel. *denotes the use of the IR based frame selection method instead of uniformly selecting the frames to be manually segmented.

Sequence	*Resolution*	*No. frames*	*f.p.s.*	*Manual frames*	$\bar{t}$	$\bar{t}_m$	$\bar{\epsilon}$
Lady Eating Apple	640 × 360	100	15	10	5.4s	2m	0.77%
Quadbike	720 × 576	50	15	5	8.6s	45s	0.22%
Polo	720 × 576	100	15	10	6.0s	1m 40s	0.52%
Oktoberfest	640 × 480	144	15	14	7.0s	25s	0.47%
Quadbike*	720 × 576	50	15	8	9.0s	45s	0.18%
Polo*	720 × 576	100	15	17	7.2s	1m 40s	0.38%

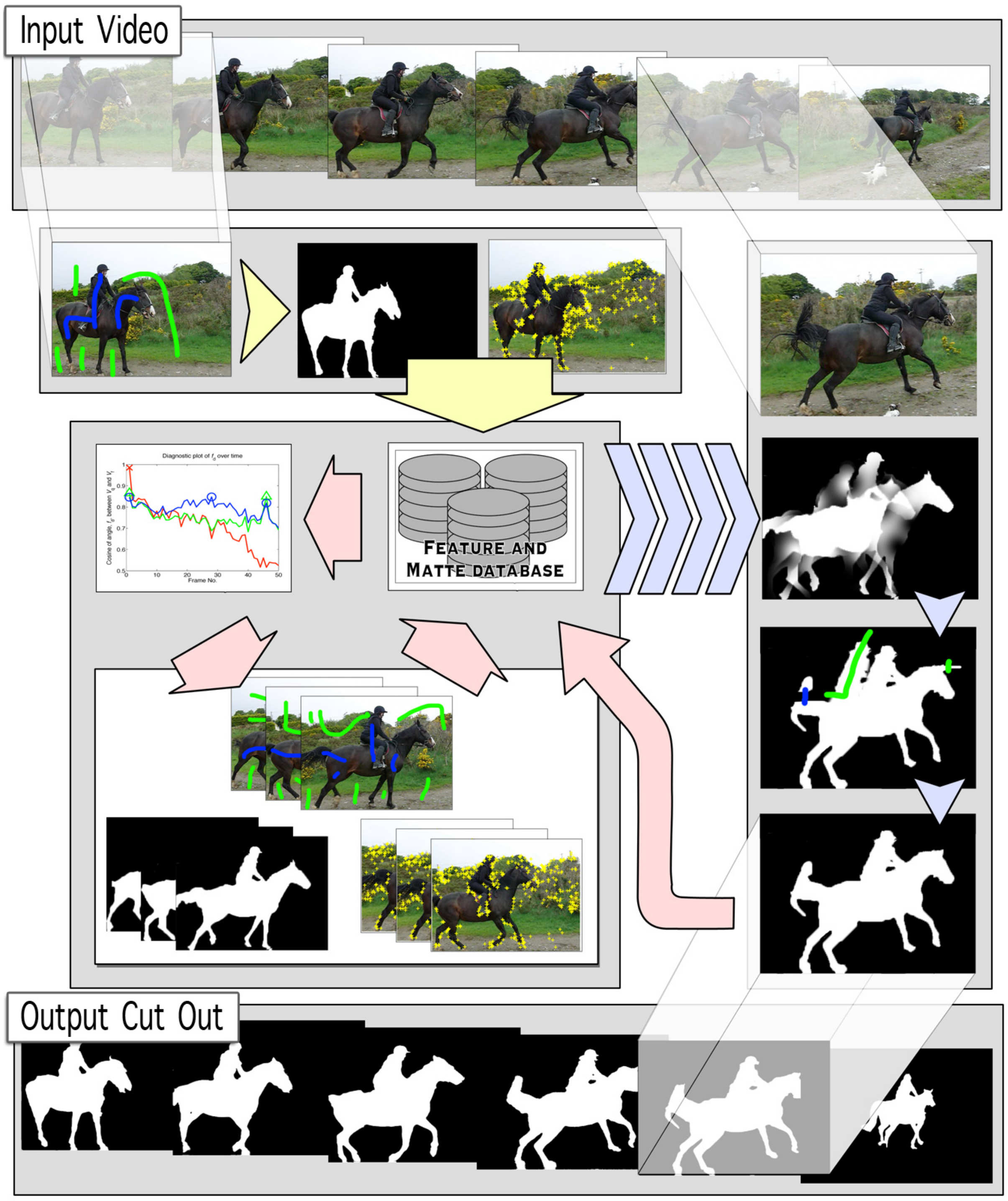

Input Video
Diagnostic plot of f_c over time
Frame No.
Feature and Matte Database
Output Cut Out

Figure 6.10: Example of typical user interaction involved in taking an input video (top), and segmenting the object of interest throughout the sequence (bottom). Details of the actions involved are described in Section 6.3.6.

The information-retrieval based technique of selecting frames to manually segment is not a fair way to compare results between sequences. It is not trivial to relate the object detection reliability score f_d of one sequence to another. For example, two sequences might require greatly different numbers of frames to be manually segmented to have the same mean f_d score, yet in general, it is the sequence with more manually segmented frames that achieves a better matte accuracy. For this reason, the "control" frame selection method of selecting and manually labelling (at most) every 10th frame was used for the results calculated in Table 6.1.

Example results of the "Quadbike" and "Polo" sequences using the IR diagnostic approach are presented on the last lines of Table 6.1. The frames chosen in addition to every 10th frame are those shown in examples earlier in Figures 6.2 and 6.3. Notice that the numbers of manually segmented frames are higher (8 and 17). These numbers correspond to the number of frames needed to capture all apparent instances of the object. For instance, in the case of the "Polo" sequence, an entire stride of the horse. As can be seen, the error for the sequences using the IR assisted frame selection is lower than their "every 10th frame" counterparts.

6.4.1 Statistical Evaluation

When comparing user created ground truth to segmentation results intended for post production use, error measures such as the Mean-Squared-Error (MSE) have limited meaning. As ground truth must be manually labelled for naturally filmed sequences, the user imparts their own subjective ideas of what pixels are considered object and background, and also the expected quality of the labelling. This makes it difficult, if not impossible, to calculate a number that objectively measures how good the results of a segmentation are. Instead, it is more meaningful to say whether the results from the system differ from results that the average professional user would manually extract.

To evaluate this, ground truth is needed for each frame of every sequence from a number of skilled users. Considering that a frame of video could easily take a minute to segment well, the manual effort required for many users to segment even a couple of sequences is unreasonable. Instead, for a *single* chosen frame in each sequence, skilled users are asked to extract the ground truth. To ensure a reasonable level of quality, the users were asked to label the object to the point where a composite of the object onto a novel background using their still frame mattes appears realistic. The author then extracts ground truth for *every* frame of each sequence will be extracted by the author and used as a reference. The idea is to first show that the author's ground truth is sufficiently similar to that of the other users ground truth for the selected frame of each sequence. If they are sufficiently similar for that one frame, it is assumed that all of the author's reference ground truth frames for all sequences are representative of typical user's ground truth. Using the author's ground truth as reference, the automatic segmentation results can be tested to see whether the differences between algorithm and reference mattes are typical of differences between mattes from different users. Before beginning the evaluation, the statistical methods for ground truth comparison are defined.

To get an idea of how differently users select objects in an image, a frame from the sequence to be tested is manually segmented by N different users, in this case $N = 7$. The difference ϵ_u between a two binary ground truth mattes (from users i and j) G_i and G_j, indexed by the set of pixel locations $\mathcal{X}$, is given by $\epsilon_u(i,j) = \sum_{\mathbf{x}\in\mathcal{X}} |G_i(\mathbf{x}) - G_j(\mathbf{x})|$. The set of differences between user mattes is given by comparing each of the N user mattes to the other mattes, for example, a matte from user i, is compared against all mattes from users $\{i+1,\ldots,N\}$, the matte from user $i+1$ is then compared to mattes from users $\{i+2,\ldots,N\}$, and so on, resulting in $\frac{N(N-1)}{2}$ values of ϵ_u. Similarly, the difference ϵ_s between the author's ground truth matte G_s and another user i's matte G_i for the single frame is given by $\epsilon_s(i) = \sum_{\mathbf{x}\in\mathcal{X}} |G_s(\mathbf{x}) - G_i(\mathbf{x})|$. For a sequence of F frames, the difference ϵ_g between the mattes automatically generated by the system G_a and the author's ground truth mattes G_g for frame $f \in \{1,\ldots,F\}$ is given by $\epsilon_g(f) = \sum_{\mathbf{x}\in\mathcal{X}} |G_a^f(\mathbf{x}) - G_g^f(\mathbf{x})|$.

In a statistical sense, two hypotheses need to be tested for each sequence. Firstly, it is asserted that the ground truths mattes of the author are not significantly different from ground truth mattes of other users, in terms of the null hypothesis, $H_0^1 : \bar{\epsilon}_u = \bar{\epsilon}_s$. Secondly, given that the author's ground truth is representative of typical user mattes, it is asserted that the automatic segmentation results are not significantly different from the author's ground truth, $H_0^2 : \bar{\epsilon}_g = \bar{\epsilon}_u$. However, H_0^2 only asserts that the distributions of ϵ_g and ϵ_u are sufficiently similar. In some cases, the distribution of differences between the automatically generated mattes and the author's ground truth mattes may be significantly lower. Therefore, an additional null hypothesis is tested, $H_0^3 : \bar{\epsilon}_g < \bar{\epsilon}_u$. Paired Student's "t" tests are performed, and evaluated at the $\alpha = 0.05$ significance level. The results for the "Polo", "Lady Eating Apple" and "Quadbike" sequences are presented in the following table.

Sequence	H_0^1	p_0^1	H_0^2	p_0^2	H_0^3	p_0^3
Lady Eating Apply	accept	0.47	reject	0.0	accept	1
Polo	accept	0.14	accept	0.1	reject	0.052
Quadbike	accept	0.739	reject	0.018	accept	0.99

The first interesting result to note, is that the null hypothesis H_0^1 asserting that the author's ground truth mattes are sufficiently similar to other user's ground truths is accepted. Had this been rejected, the ground truth manually created by the author would not be representative of typical users, and useless as a basis for comparison. The next important result is that the null hypothesis H_0^2 asserting the differences between the automatic and author's ground truth mattes are consistent with the differences between typical user mattes is accepted for the "Polo" sequence. The reason that H_0^2 is rejected for the "Lady Eating Apple" and "Quadbike" sequences is that the differences between the automatic and author's ground truth mattes are significantly less than differences between typical user mattes, therefore accepting the null hypothesis H_0^3. Although $\bar{\epsilon}_g$ is lower than $\bar{\epsilon}_u$, this does not necessarily mean that the automatic segmentation results are better than mattes that a typical user would create. However, had $\bar{\epsilon}_g$ been significantly higher than $\bar{\epsilon}_u$, this would have indicated very poor mattes, and a complete failure of the segmentation system. Given that either H_0^2 or H_0^3 is accepted for all sequences, the results of the automatic segmentation algorithm are considered not to be significantly different from mattes that a typical user would manually create for the same sequences.

6.4.2 Visual Evaluation

To appreciate the visual results of this chapter, the reader is directed to the object segmented videos on the accompanying DVD, as detailed in Appendix D.2. The cut out results of each of the sequences by selecting every 10th frame as a key-frame. For each sequence, the original video is shown, followed by the sequence of the author's ground truth to give an idea what the system input actually looks like. The automatic binary segmentation results from the presented algorithm are then shown. As with other segmentation systems, the object cut out is then supplied as input to a non-binary matting tool ("Nuke" [169]), and composited onto a blue background. A comparison of the cut out results for the "Quadbike" and "Polo" sequences is then presented, highlighting the differences between selecting every N^{th} frame as a key-frame, and using the IR diagnostic approach to selecting optimal key-frames to select.

A visual comparison of example frames from the four sequences is presented in Figures 6.11, 6.12, 6.13 and 6.14. As with the results in Table 6.1, the mattes are not manually touched-up after the automatic matte generation. In these figures, the blue background compositing was performed by using the mask as input to the non-binary matte tool in Nuke [169]. A comparison between automatic feature-based segmentation and manual segmentation algorithms is shown in Figure 6.19. An interesting comparison between using features and using multi-scale motion estimation to propagate user mattes is shown in Figure 6.18. Examples of frames segmented from additional videos are shown in Figure 6.15. The reader is directed to the DVD to view the complete versions of these clips, details are in Appendix D.2.

To illustrate the accuracy of the refined object matte close-ups of frames from the "Lady Eating Apple" and "Quad-bike" sequences are shown in Figures 6.16 and 6.17. The close-up of Figure 6.16 highlights some interesting properties of the presented algorithm. The scene from "Lady Eating Apple" contains a mixture of dense, detailed plant foliage and a "head-and-shoulders" with little texture. This detailed scene is reasonably difficult for traditional segmentation features, such as colour or edges. However, stable feature points are generally detected around highly textured regions. As the performance of the algorithm is related to the number of feature points, the more detailed and complex the scene, the greater the matte accuracy. Notice in Figure 6.16 that the detail of the leaves is preserved, particularly in the branch occluding the subjects face. The close-up of the "Quad-bike" matte in Figure 6.17 also illustrates how difficult regions are successfully segmented, such as the sparse frame of the quad-bike.

The results of the “Oktoberfest” sequence shown in Figure 6.14 are interesting. The shot begins by a left-to-right pan of the crowd, then focuses on the subject for a while, and resumes the pan. The manually segmented frames containing the subject were taken from the middle of the shot. Notice in the top and bottom rows of Figure 6.14 (corresponding to the beginning and end of the shot), the subject is not in the shot, and is correctly not segmented. As the automatic segmentation process is based upon matching features between images, no explicit object detection is needed. With the presented algorithm, the object is free to enter and exit the shot, and will only be segmented when it is in the frame.

The segmentation results in Figures 6.20 and 6.21 show examples where the algorithm fails. In Figure 6.20, regions of the rider’s arms disappear. In this case, the video frame is the last one in the sequence and is relying on the appearance of the arm to remain unchanged since the manually extracted matte information from 10 frames prior. There is also the problem of foreground and background regions with similar colours merging, such as the handlebars. As these regions are relatively texture-less, few feature points are found there, making the segmentation more reliant upon the spatial prior than the data energy of Equation 6.3.

Looking at the results shown on the accompanying DVD (Appendix D.2), it can be seen that the quality of the user input (i.e. selections of the ground truth frames) is far from ideal. Apart from the lack of detail in many areas, when played back as a video, the ground truth appears to be highly temporally inconsistent. This is because each frame is labelled independently, causing the subjective object boundary to hop wildly between frames. It is interesting to see that in many cases the automatic segmentation results appear sharper, and often capture the object better than the user labelling.

6.5 Discussion

The goal of this work is to reduce the effort required by the artist to create an object cut out throughout a video sequence, allowing the user to segment longer videos, with higher accuracy. The results of Table 6.1 show that the manual effort required to segment the frames can be dramatically reduced by nearly an order of magnitude for comparable levels of quality. In addition, the diagnostic approach to selecting frames to manually segment provides another dimension to user-interaction, removing a lot of trial and error from the process, and making interaction more predictable. This simple tool extends the range of the automatic stage of segmentation, making object matting of longer sequences far easier. From the visual results shown in Section 6.4.2 (and the accompanying clips on the DVD in Appendix D.2), the quality of the segmentations without additional user corrections are already of a very useable standard. Visually, it can be seen that relatively few corrections would be needed to bring these clips to a post production standard. The objectives of this work, to allow fast, high quality matte creation over longer and more detailed sequences, have been achieved.

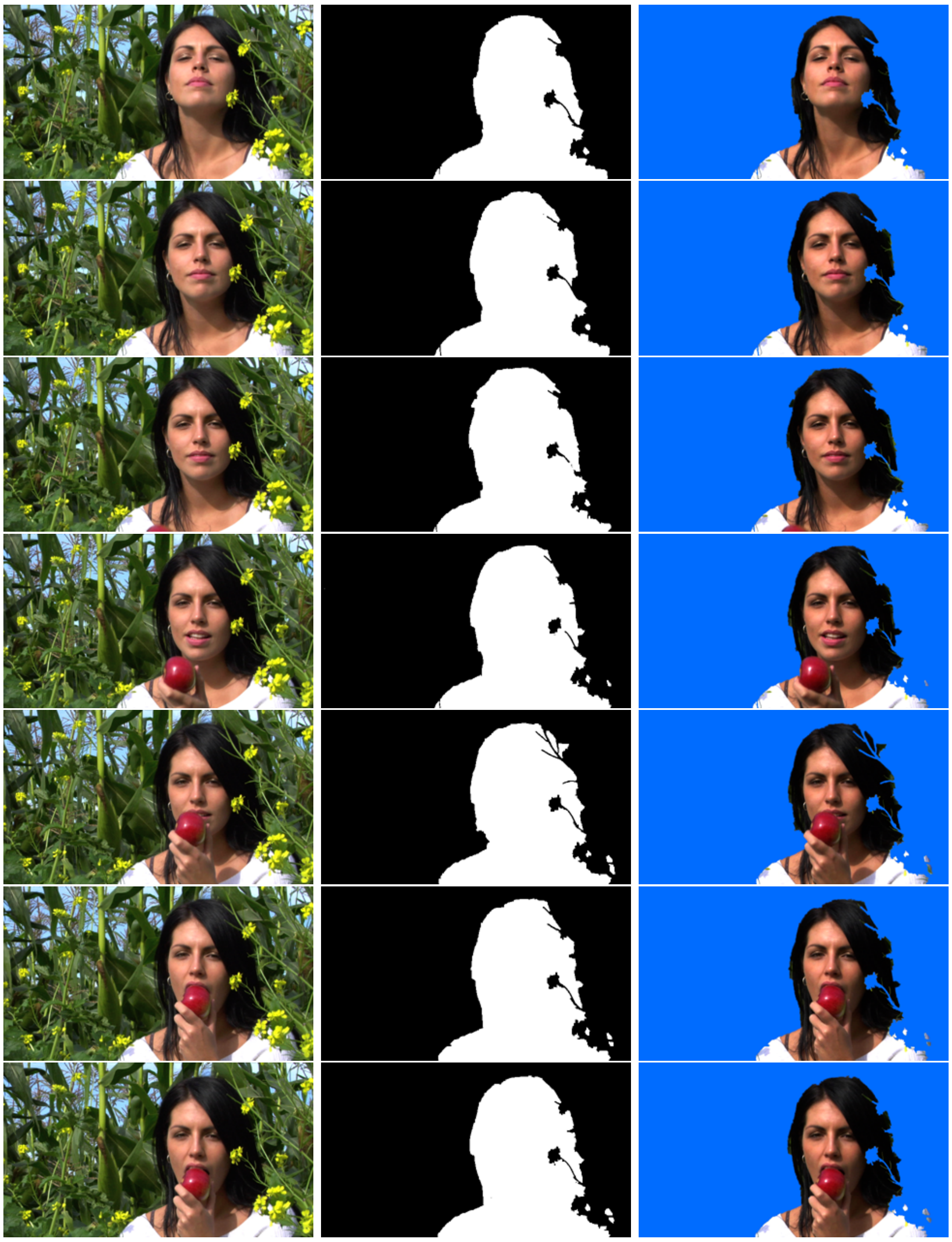


Figure 6.11: Results of the "Lady Eating Apple" sequence. Every 10th frame of the original video is manually labelled and supplied to the algorithm as input. Taking some example video frames to be segmented (left), mattes are automatically extracted by the system (centre) and composited onto a blue background (right).

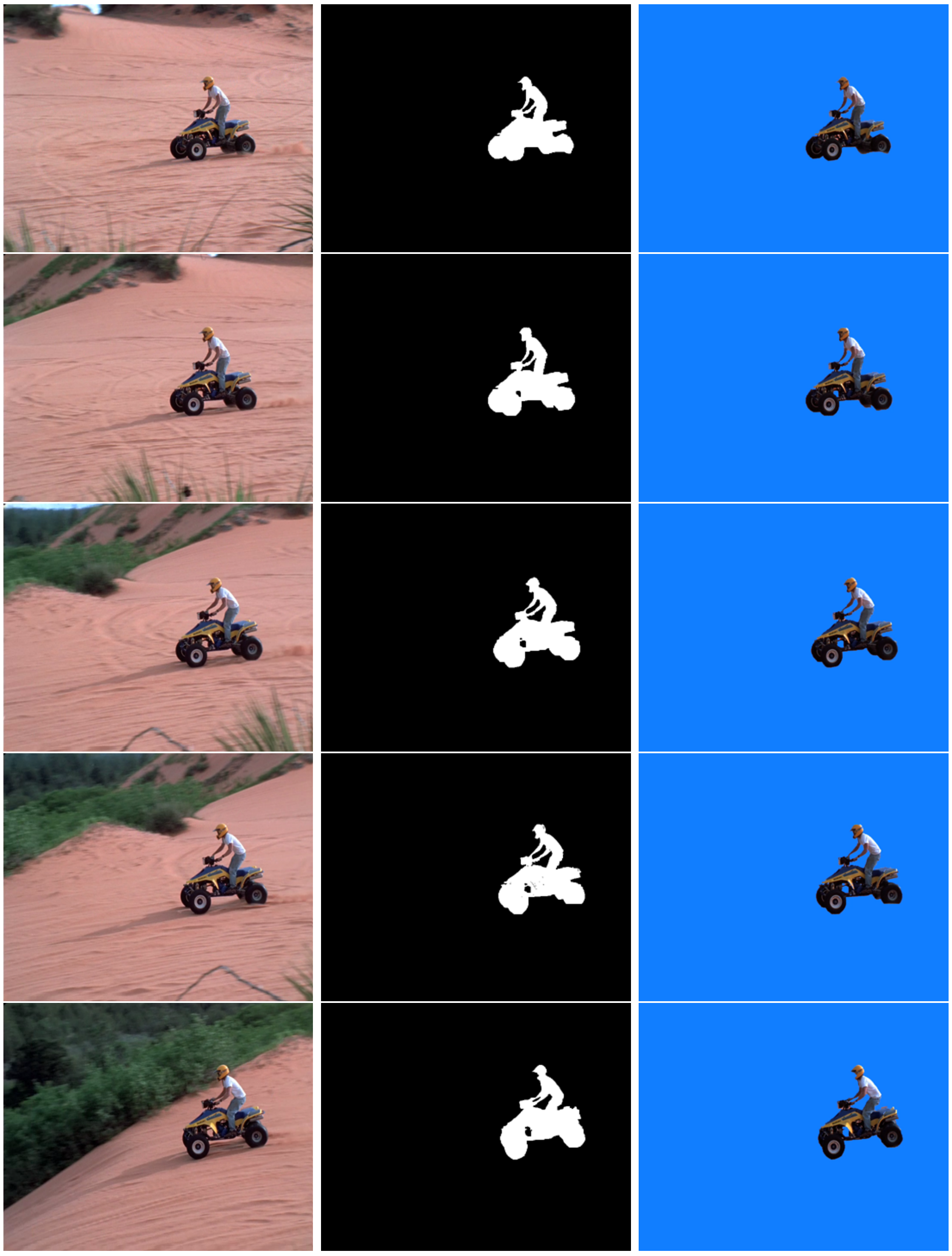


Figure 6.12: Results of the "Quad-bike" sequence. Every 10th frame of the original video is manually labelled and supplied to the algorithm as input. Taking some example video frames to be segmented (left), mattes are automatically extracted by the system (centre) and composited onto a blue background (right).

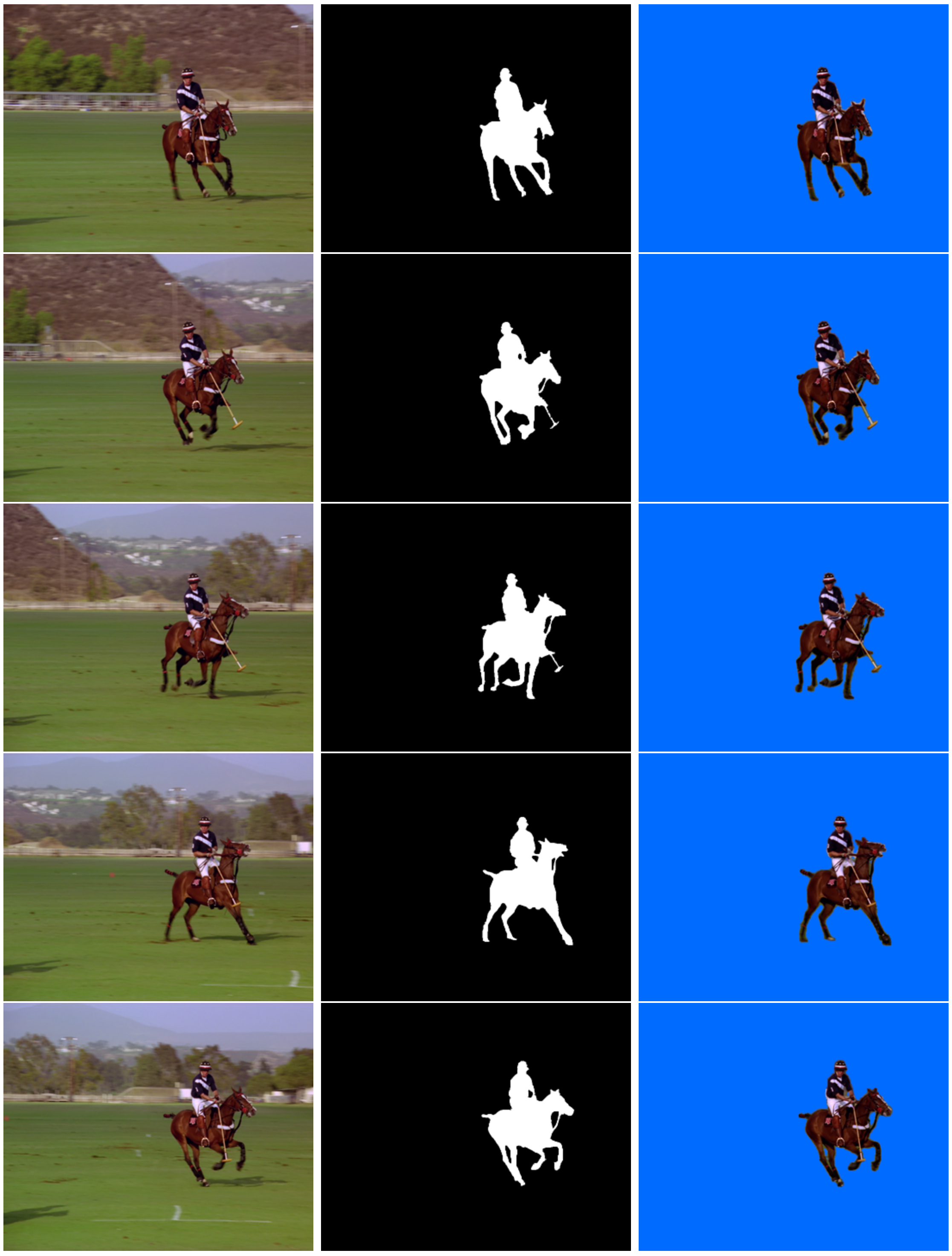


Figure 6.13: Results of the "Polo" sequence. Every 10th frame of the original video is manually labelled and supplied to the algorithm as input. Taking some example video frames to be segmented (left), mattes are automatically extracted by the system (centre) and composited onto a blue background (right).

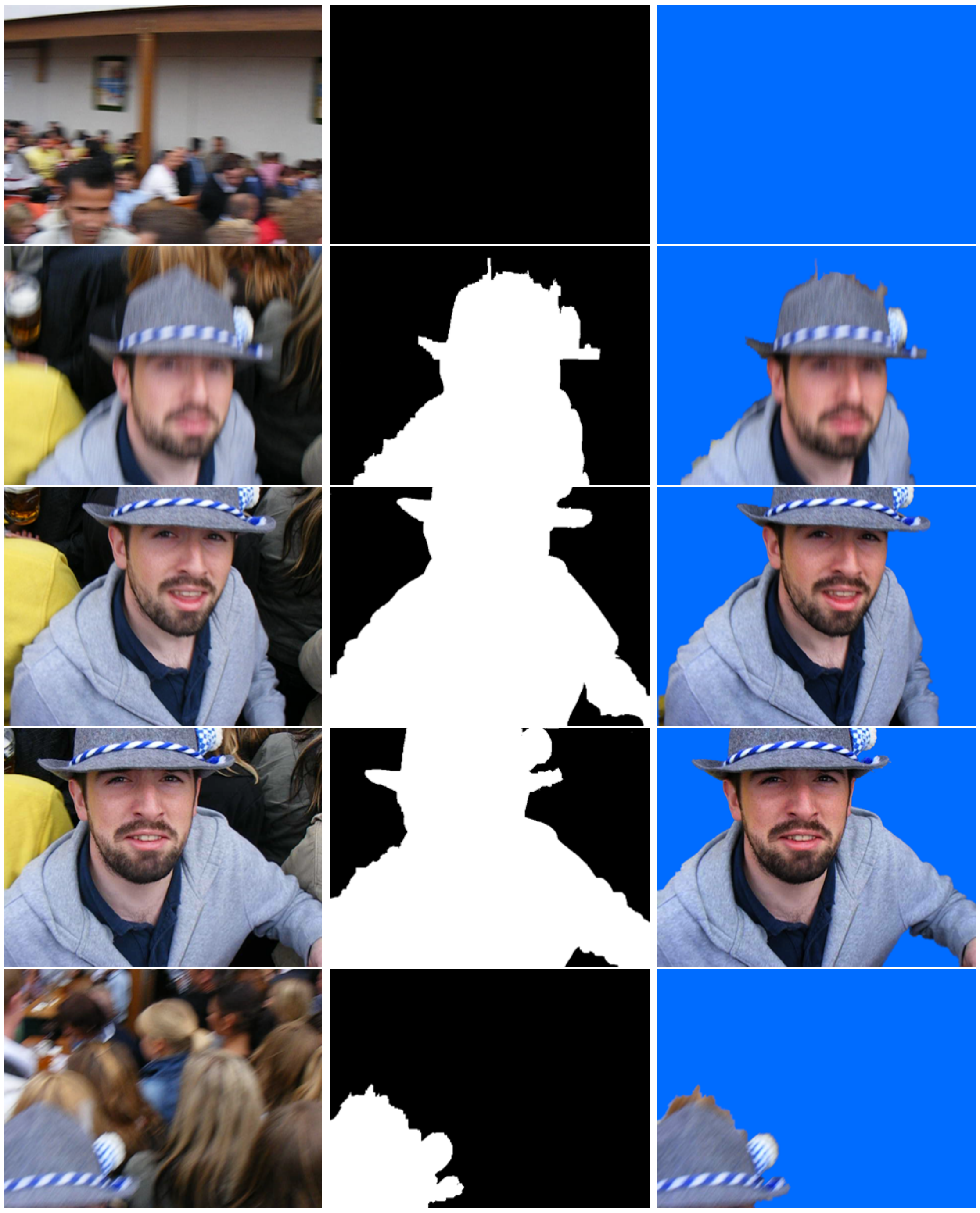


Figure 6.14: Results of the "Oktoberfest" sequence. Every 10th frame of the original video is manually labelled and supplied to the algorithm as input. Taking some example video frames to be segmented (left), mattes are automatically extracted by the system (centre) and composited onto a blue background (right).

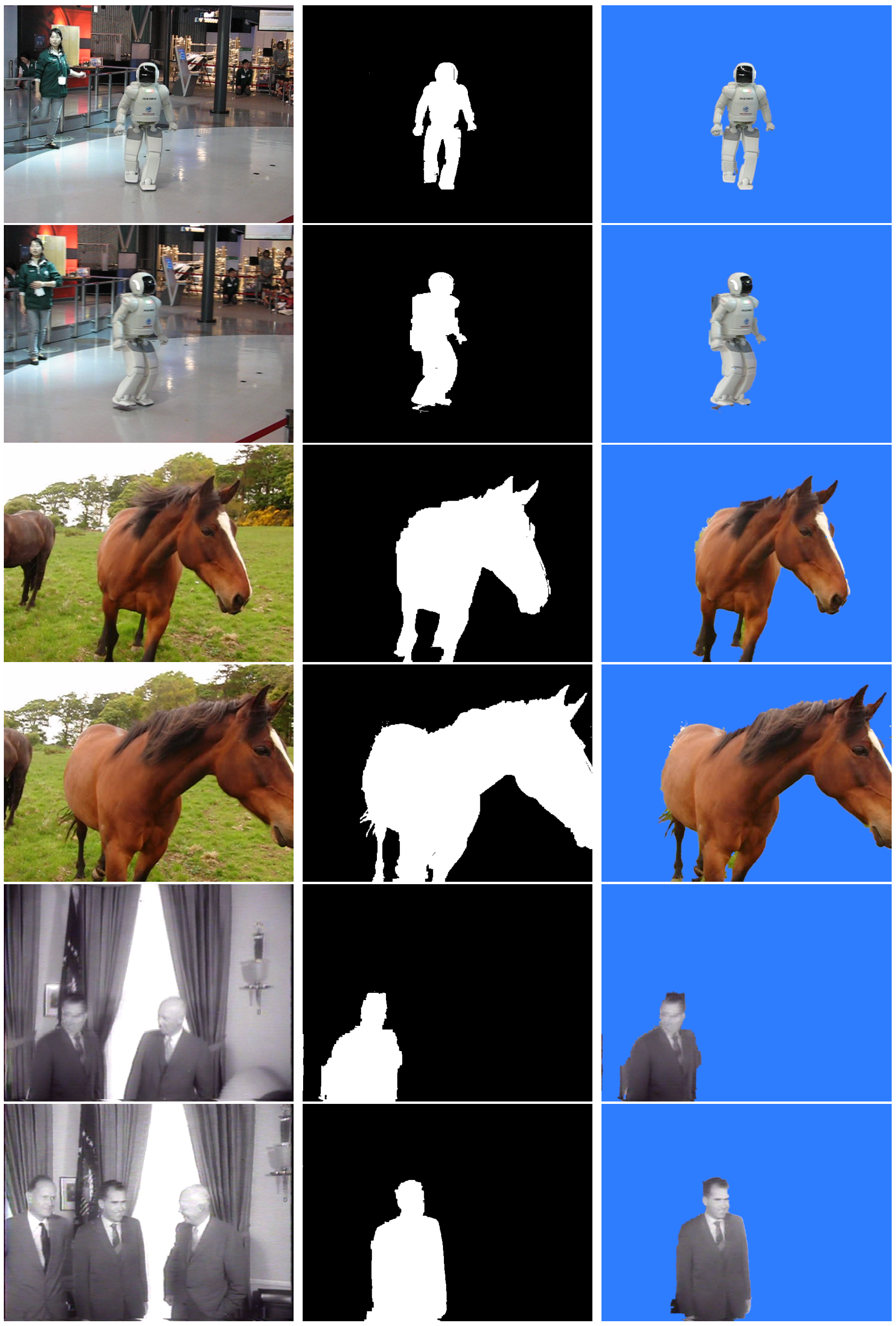


Figure 6.15: Additional challenging video segmentation results. The "Asimo" and "Horse Turn" (top and middle) clips were shot using simple point-and-shoot cameras, with significant background clutter, similar foreground and background colours ("Asimo") and image noise introduced by rain ("Horse Turn"). The "Nixon" (bottom) clip is severely degraded, exhibiting large interlacing artefacts, motion blur, sudden changes in focus, poor contrast and no colour information. In spite of these difficulties, the presented system performs reasonably well. The reader is directed to the DVD to view the complete versions of these clips, details are in Appendix D.2.

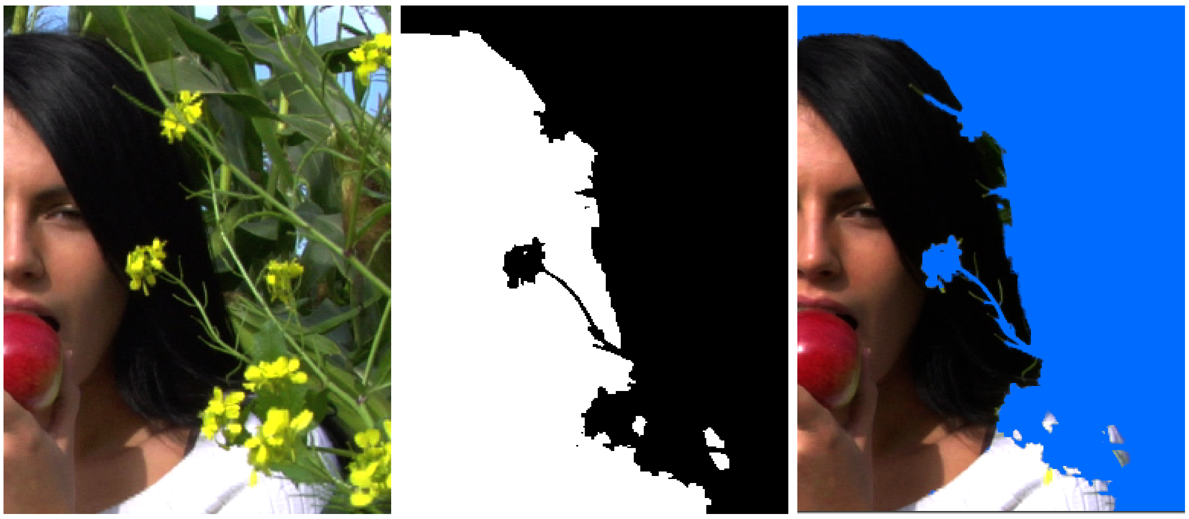


Figure 6.16: Close-Up of results from the “Lady Eating Apple” sequence. From left to right; original, matte from algorithm, non-binary matte composite onto blue background. Despite the intricate shapes of the plant’s branches, an accurate matte can be found. Notice however, that some finer plant blanches were not segmented. During the manual segmentation of the object in other frames, the same fine structures were not selected by the user, and so have not been segmented by the algorithm. Whether strong fidelity to the user mattes is good or not is a matter of taste.

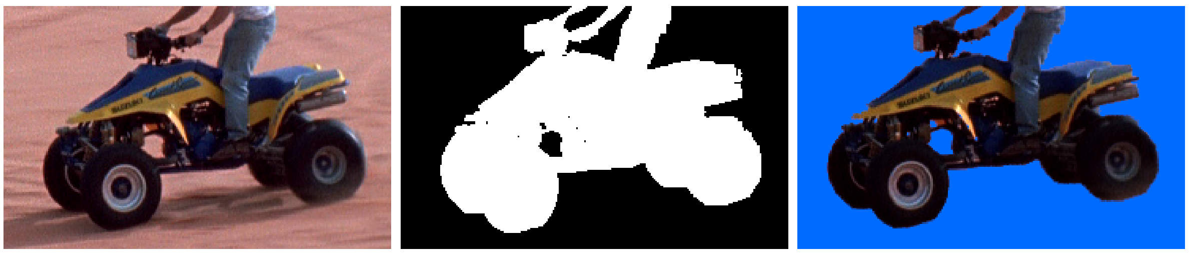
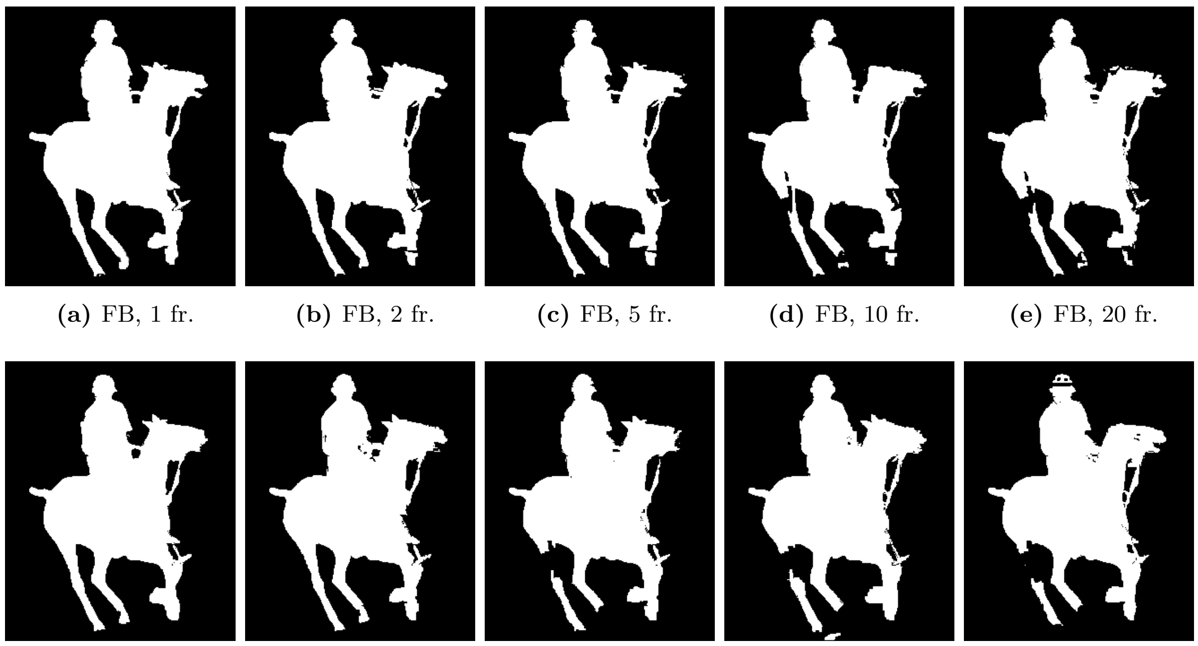


Figure 6.17: Close-Up of results from the “Quad-bike” sequence. From left to right; original, matte from algorithm, non-binary matte composite onto blue background. Two things are interesting here; the see-through parts around the frame of the quad-bike are preserved, and foreground / background boundaries with low contrast (such as the shadows and wheels) are correctly and accurately segmented.



(a) FB, 1 fr. **(b)** FB, 2 fr. **(c)** FB, 5 fr. **(d)** FB, 10 fr. **(e)** FB, 20 fr.

(f) ME, 1 fr. **(g)** ME, 2 fr. **(h)** ME, 5 fr. **(i)** ME, 10 fr. **(j)** ME, 20 fr.

Figure 6.18: Comparison between using the feature-based ("FB", top-row) and motion-estimation ("ME", bottom-row) to propagate mattes from various temporal distances. In the figures shown, frame 25 of the "Polo" sequence is being automatically segmented from two manually supplied mattes, one before and one after frame 25. For example, "5 fr." means that user mattes from frames 20 and 30 (i.e. 25 ± 5) are used as the source mattes. The motion estimation algorithm is the multi-resolution version of the Horn & Schunk [74] optic flow algorithm, with parameters tuned specifically for this sequence to allow for meaningful comparison. One of the interesting things to note is that ME gives smoother, more coherent mattes, i.e. less label "noise". However, it can be seen that ME results degrade more rapidly than FB as the temporal disparity increases. This is particularly noticeable in the horse's back legs (substantial parts missing from 5 fr. onwards), and the rider's helmet. This is expected ME behaviour; the "small motion" assumption that the optic flow algorithm relies upon for convergence breaks down as the actual motion becomes too great. It is interesting to see that the difference between FB propagation results for frame disparities "10 fr." and "20 fr." is relatively low. In practice, results for this frame from source matte disparities greater than 20 frames will provide similar results for this sequence, as it is likely that any two frames capture this minimum amount of "horse-ness". (In fact, at around 24 or 25 frame disparity there will be an increase in quality for this sequence, as the disparity corresponds to the horse's walk-cycle period). As shown, to improve segmentation quality, the FB frame disparities should be less than 10 frames. To achieve similar quality with ME, a disparity of 2 frames is required.

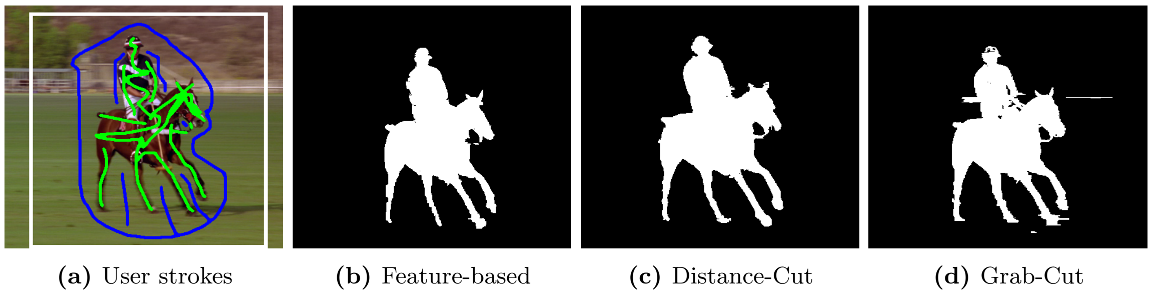

(a) User strokes (b) Feature-based (c) Distance-Cut (d) Grab-Cut

Figure 6.19: Comparison of the presented algorithm to other segmentation schemes. The feature-based segmentation result in Fig. 6.19b shows the automatically segmented frame no. 2 of the "Polo" sequence, which was calculated by propagating user mattes from frames 10 and 20 of the sequence. The Distance-Cut segmentation in Fig. 6.19c was calculated on frame 2 only, using the manual Fg. & Bg. strokes shown in Fig. 6.19a (green & blue resp.). Similarly, the Grab-Cut result was calculated using the white input rectangle shown in Fig. 6.19a, using only a single automatic segmentation pass. Notice that the results of the automatic feature-based segmentation are not significantly different from the manual Distance- and Grab-Cut segmentation algorithms, and in the case of Grab-Cut, slightly better. The similarity between the Distance-Cut and automatic feature prop. results is not surprising, as the Distance-Cut algorithm was used to create the manual labellings on frames 10 and 20.

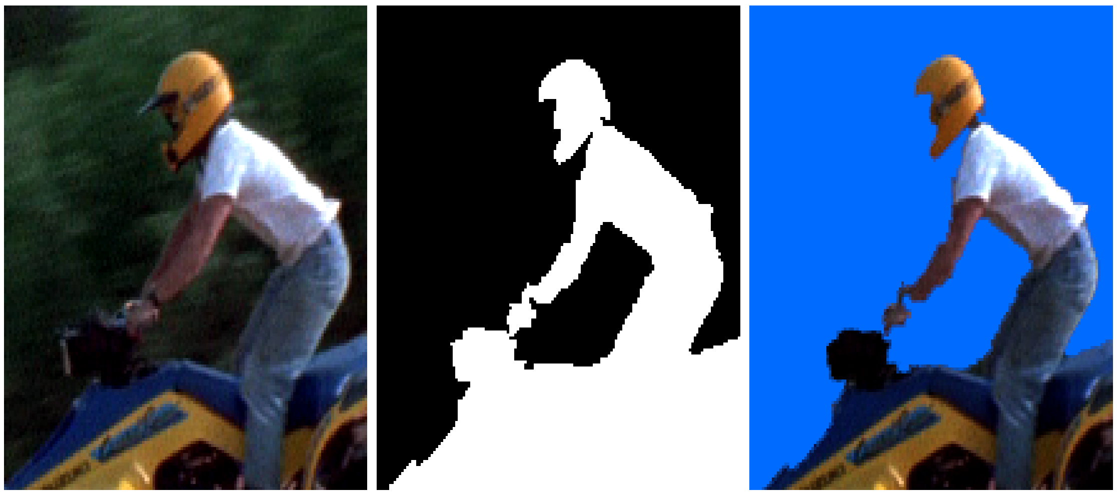


Figure 6.20: Example of the algorithm failing on the "Quad-bike" sequence. The lack of contrast between the background and parts of the rider and quad-bike have caused parts of the face, helmet, arms and handlebars to become "eroded". In cases such as this, the artist would be required to touch-up the matte.

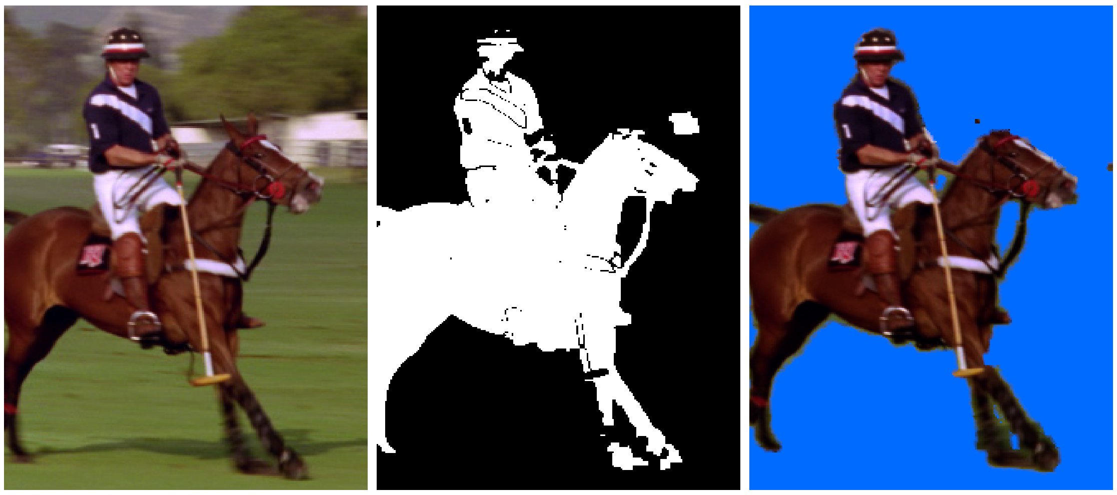


Figure 6.21: Example of the algorithm failing on the "Polo" sequence. The low number of feature correspondences between this frame and those with user-supplied mattes (mostly due to motion blur) has caused a number of missing regions on the horses legs, ears. Visible in the centre image, a number of incorrect feature matches have introduced blotches falsely accepted as object, and caused part of the helmet and jersey to be falsely labelled as being background. Notice that the non-binary matte (right) can compensate for a lot of mistakes, but the result is still not perfect. For example, the ears are still missing, due to low colour contrast. Again, in this case the user would need to manually touch-up the matte.

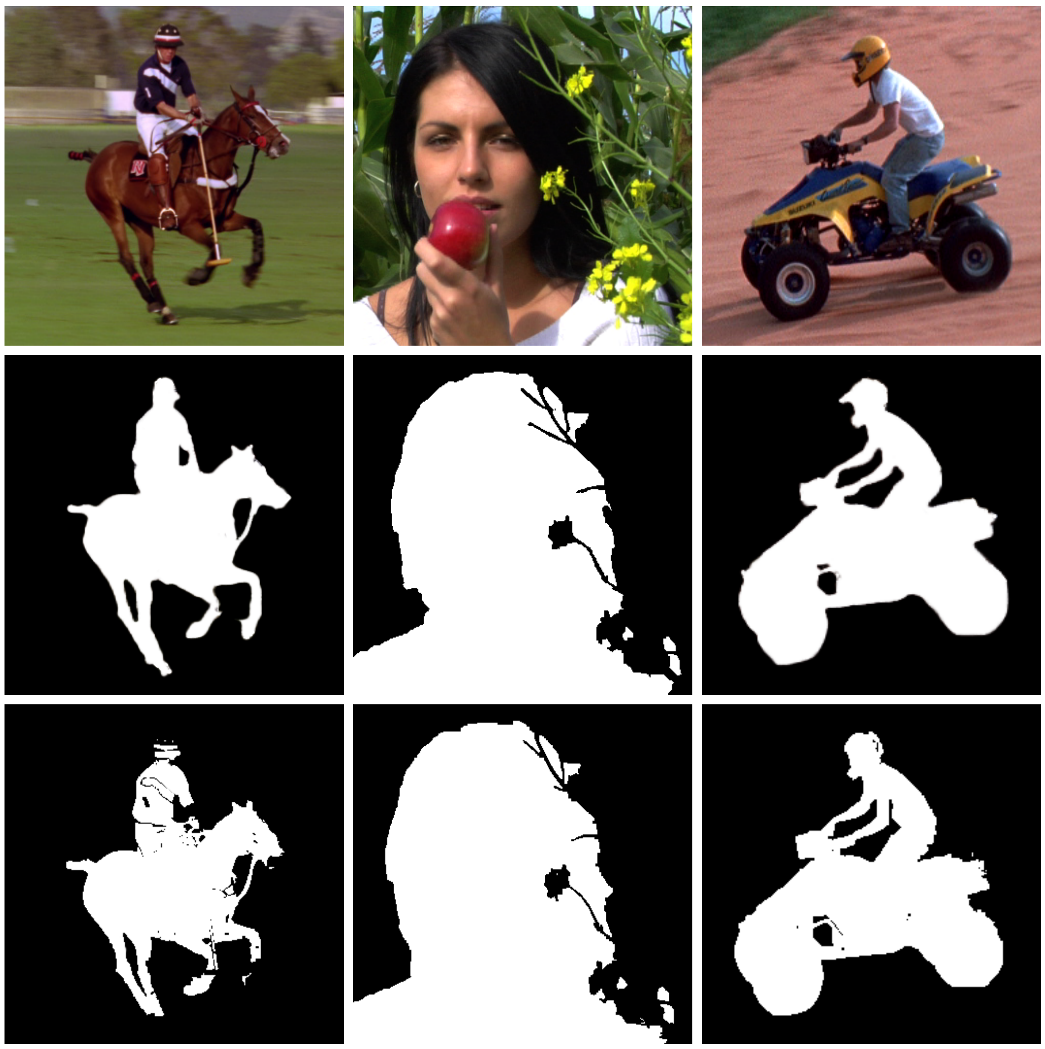


Figure 6.22: Comparison of close-ups of original (top row), ground truth (middle row), and algorithm output (bottom row) from (left to right) frames, 40, 31 and 35 of the “Polo”, “Lady Eating Apple” and “Quad-bike” sequences respectively. The selected frames exemplify some of the strengths and weaknesses of the system. For example, the output frame of the “Polo” sequence (bottom, left) exhibits “missing” matte data, particularly around the rider’s helmet, left arm, and horse’s front left leg. This is a direct result of not having enough feature matches to propagate matte information. In the case of the “Lady Eating Apple” frame (bottom, centre), the highly detailed scene allowed more features to be correctly matched, resulting in a matte that looks very similar to the ground truth (middle, centre). The “Quad-bike” frame (bottom, right) is also very close to the ground truth, and in some places has successfully identified not-object pixels not present in the ground truth, for example, near the bike’s chassis and frame. However, similar foreground and background colours have introduced errors. For example, to the rear of the bike, and the front of the rider’s helmet.

As mentioned at the beginning of this chapter, elements of this work are similar to the Video Snap-cut segmentation system developed in parallel by Bai et al. [11]. The Video Snap-cut system propagates detailed matte information from user key-frames to unlabelled frames. The presented system takes a similar approach, but differs in two significant ways.

Matte propagation In Video Snap cut, colour and shape information from object boundary regions are propagated forward into subsequent frames to be segmented, using optical flow to identify corresponding regions between frames. The idea is then to favour the use of colour to segment the region when the colour distributions are well separated, falling back on the propagated boundary shape when the distributions are not well separated. However, in the presented system, it is the shape information alone from many inter-frame feature matches that are used to generate the object and background data energies. Note that the data energy for a single pixel may be calculated from the propagation of many correspondences, across many temporally disparate mattes, leading to very rich data likelihoods in regions of dense feature correspondences.

Error weighting Similar to Video Snap-cut, the presented system attempts to identify and mitigate errors in the dominant feature space. Instead of falling back on a different feature cue when a feature match is poor, the contribution of the match on the data energies is weighted down by the difference in image appearance (the ΔI term in Equation 6.3). However, the use of complementary cues to compensate for errors is interesting, and will be investigated in future work.

A few problems have been identified with the presented feature-based approach to segmentation. The Achilles heel of any segmentation scheme is the feature space used. For example, colour segmentation fails if the colour in the image is poor, contour segmentation fails if the edge information is poor, and motion segmentation fails if the apparent motion is poor. The main problem with feature-based segmentation is of course, when feature information is poor. Texture-less regions, motion-blur and non-uniform object deformation make it difficult to detect reliable features. These conditions are exemplified by the horse's leg in Figure 6.13. Although texture-less regions can be compensated for during refinement of the propagated partial mattes, there are some situations where the presented feature-based segmentation will fail. It may be possible to include additional feature cues, as shown by Price et al. [136], or by including exploiting user information further to improve segmentation performance and quality.

Another issue is that each frame to be automatically segmented is processed independently of consecutive frames. For example, following the manual segmentation stage, the remaining frames can be processed in any order; no temporal consistency is enforced. This has the effect of pixels "popping" in and out of the automatically generated mattes over time. It is unclear whether this is introduced by the user labelling, or whether it is simply an effect of processing frames independently of each other. The Video Snap-cut and LIVEcut systems also exhibit similar popping effects to varying degrees. In practice, the non-binary matting stage will generally be able to compensate and reduce the apparent effects. However, it makes sense to incorporate information between consecutive frames to produce better mattes before applying non-binary matting.

Chapter 7

User Assisted Feature Matching[1]

Feature matching is a vital stage in many image processing applications. Finding accurate correspondences is made difficult by phenomena such as occlusions, non-rigid deformations, motion blur and more. This chapter posits that some scenarios simply do not have enough information for an accurate automatic solution. Although many applications are required to be automatic, there are others that can benefit from being semi-automatic, allowing the user to provide assistance to areas where the system is failing. Good examples of this exist in the media post-production world, such as multi-view scene reconstruction, sparse-to-dense disparity estimation from view matching, image mosaic'ing (digital panoramas), or even motion estimation. The presented work describes how to incorporate user-assistance into a Bayesian feature matching framework. By adding user information in the form of intuitive Bezier curves, difficult regions can be matched with the same accuracy as easier to match areas. The presented system uses a simple optimisation scheme, giving the user real-time interactive control over the corrected matches.

7.1 Introduction

Feature matching is an important part of many image processing tasks, such as image registration, object tracking, multi-view scene reconstruction and depth estimation from stereo image pairs. The accuracy of these higher level tasks depends on the accuracy of matching features. In real world images, feature matching is made difficult by non-rigid object motion, blurring and poor textural content (*pathological content*). There is often not enough information to automatically reject incorrect matches or propose better alternative matches. Given that user intervention is typical in media post-production, an interactive system is proposed to allow the user to encourage better feature matches in difficult regions, examples of which are shown in

[1]Results from this chapter have been published as "User-Assisted Feature Correspondence Matching" by Dan Ring and Anil Kokaram, in *IEEE European Conference on Visual Media Production (CVMP)*, London, UK, November 2009.

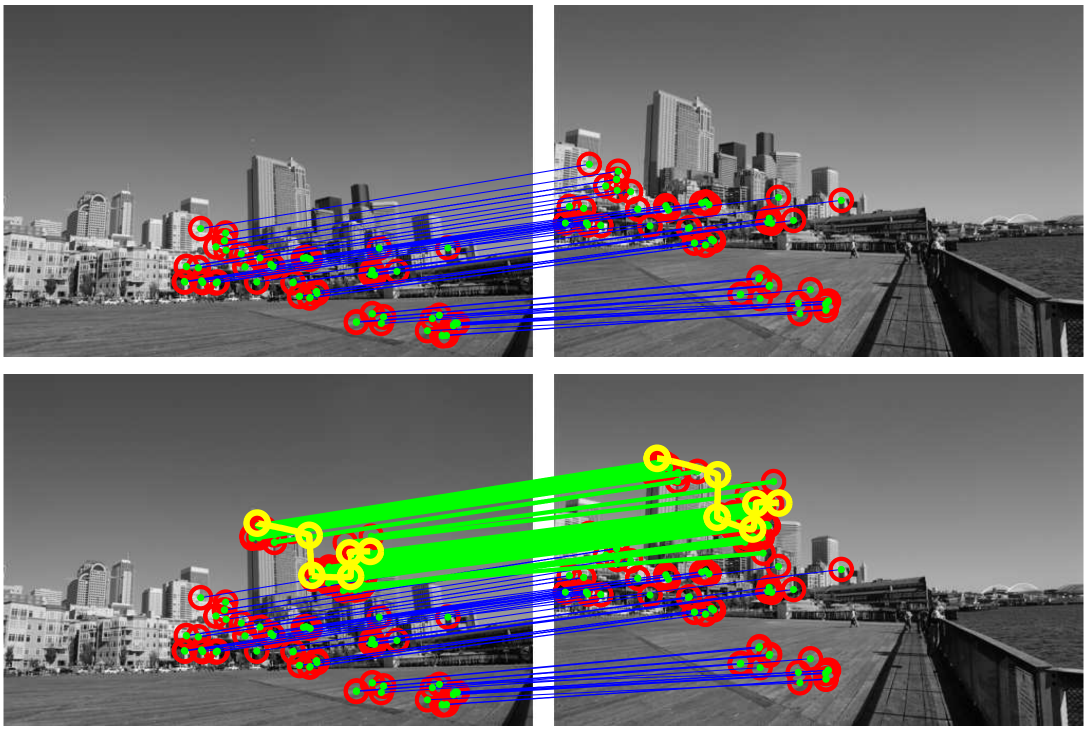


Figure 7.1: Example of a registration task. Top: A transformation is estimated between the two images using RANSAC, and the best correspondences are shown (blue). The model did not capture the perspective distortion and missed possible matches on a number of buildings. Bottom: two simple curves are provided by the user as a rough guide (yellow), and the previously overlooked matches are recovered (green).

Figures 7.1 & 7.2.

As discussed in Chapter 2, given sets of *sparse* features in two frames (such as Harris-Laplace corners [115, 70] or SIFT features [105]), previous work concentrates on finding correct correspondences between them. Most matching strategies compare local "descriptors", around the sites of the features. A low descriptor distance between a pair of features indicates a likely match. Simple heuristics such as "nearest-neighbour" and "ratio-testing" [105] attempt to match based solely on descriptor distances, disregarding the spatial context of the features. It is sensible that incorporating spatial feature context in some way, can improve feature matching performance. For example, by estimating piece-wise transformations of subsets of features [172], or requiring that at least N similar feature correspondences lie in the vicinity of a potential match [161].

Recently, stereo imaging and multi-view object reconstruction are becoming more popular in post-production scenarios. Many of these applications greatly depend on the accuracy of feature correspondences. When the feature matching stage fails, digital artists paint over the problematic areas introduced by incorrect feature correspondences, such as touching up missing

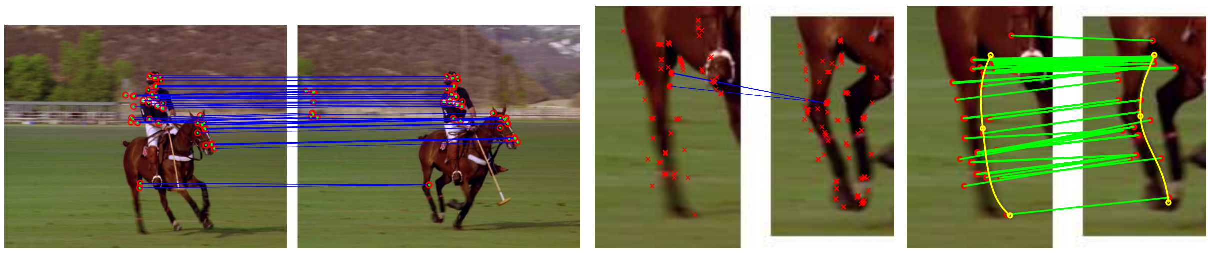


Figure 7.2: Some scenes do not have enough information to provide accurate feature matches. Left image pair, a selection of the strongest correspondences is given between the two frames in (blue), using the default matching scheme described in Section 7.3. Regardless of parameters, correspondences are never identified between the flat untextured regions of the horse, although features do indeed exist for these areas (centre, red). Following the manual addition of 2 Bezier curves to the shape of the horse's leg in the right image (yellow), matches are encouraged and identified in the difficult regions (green).

or erroneous depth information, or manipulating the 3D points of the reconstructed object. The presented work uses an interactive method of correcting mistakes at the feature matching stage in order to reduce the amount of low-level correction needed by the artist. These sorts of semi-automatic systems have already shown that results can be dramatically improved with minimal user intervention, the prime example is the field of object segmentation [41, 10].

Torresani et al. [173] pose the problem of matching using an energy that combines spatial information with descriptor matching. Graph matching techniques are then used to solve the resulting optimisation problem. Despite improved matching performance, problematic regions (with *pathological content*) remain difficult. Figure 7.2 (left) shows an example of feature correspondences similar to those found by Torresani et al. between a pair of images in a sequence. There are no feature matches found on the horse's legs due to a combination of heavy deformation and blur. However, visually it can be seen that matches should exist in these regions. Other related work that explicitly detects non-rigid deformation [158], or similarly performs quasi-dense matching [93], relies on first finding a small set of reliable matches. Again, the number of features matched in Figure 7.2 (centre pair) is insufficient as bootstrap matches. This reaffirms the premise that some situations simply do not have enough information to provide accurate feature correspondences.

The novelty in this chapter is the development of a feature matching energy function that incorporates user information. The user begins by drawing two smooth curves roughly covering corresponding, yet unmatched regions. Feature matches are then encouraged in the vicinity of these curves. The advantage of this technique is that the algorithm is not restricted to the exact form of the curves, giving a good interface for user input. This provides an interactive method of correcting mistakes at the feature matching level. The new matching framework is presented next, and the process of user interaction is described. Results are discussed in Section 7.3.

7.2 User-Assisted Matching Framework

Following the notation of Torresani [173], consider that P' and P'' are the features detected in two images, and $A \subseteq P' \times P''$ denotes the set of possible assignments between these features. A *matching configuration* is defined by the binary-valued vector $\mathbf{x} \in \{0,1\}^A$. A potential correspondence $a \in A$ indexes an entry x_a in the vector $\mathbf{x}$. The correspondence a exists if $x_a = 1$, and does not exist if $x_a = 0$. Given a feature $p \in P'$, $A(p)$ is the set of correspondences in P'' involving p. The objective is to find a matching configuration $\mathbf{x}$ that minimises the energy function,

$$E(\mathbf{x}) = \lambda^{app} E^{app}(\mathbf{x}) + \lambda^{usr} E^{usr}(\mathbf{x}) + \lambda^{geo} E^{geo}(\mathbf{x}) \ . \tag{7.1}$$

The components of $E(\mathbf{x})$ are the feature appearance energy, $E^{app}(\mathbf{x})$, the proposed user-assisted energy, $E^{usr}(\mathbf{x})$, and the spatial consistency energy, $E^{geo}(\mathbf{x})$, each of which is described below. The scalars λ^{app}, λ^{usr} and λ^{geo} weight the contribution of the respective energy terms.

Feature Appearance, $E^{app}(\mathbf{x})$: The function $E^{app}(\mathbf{x})$ measures the dissimilarity between a pair of features by comparing the pixel neighbourhoods around the feature sites. The image regions around the sites are described by SIFT descriptors [105] to allow for high amounts of geometric variation, such as non-rigid deformations. $E^{app}(\mathbf{x})$ is therefore given by:

$$E^{app}(\mathbf{x}) = \sum_{a \in A(p)} x_a \| d(I', \mathbf{p}) - d(I'', \mathbf{q}) \|$$

where $d(I', \mathbf{p})$ and $d(I'', \mathbf{q})$ are the SIFT descriptors calculated about sites $\mathbf{p}$ and $\mathbf{q}$ of features p and $q \in A(p)$, from the image pair I' and I''.

User Assistance, $E^{usr}(\mathbf{x})$: In this energy, cubic user-defined Bezier curves encourage correspondences in difficult to match regions. Bezier curves are already used in most image and video editing and compositing tools, making them a natural choice for user input in this situation. Imagine a region corresponding between two images exists, but is unable to be matched. In this case, the user marks the region in both images with a rough Bezier curve, an example is shown in Figure 7.2 (right). The cubic Bezier curve equation parameterised by $t = [0, 1]$ is given by

$$\mathbf{B}(t) = (1-t)^3 \mathbf{P_0} + 3(1-t)^2 t \mathbf{P_1} + 3(1-t) t^2 \mathbf{P_2} + t^3 \mathbf{P_3}$$

where $\mathbf{P_0}, \mathbf{P_1}, \mathbf{P_2}$, and $\mathbf{P_3}$ are 2D control points in the image defining the shape of the curve.

Intuitively, the curves are used as a transform function, taking locations of features around the curve in image one, and projecting them onto corresponding locations about the second curve in image two. It is not expected that the projected feature locations will match exactly to features in the second image, but the idea is to use the transformation as a soft constraint to encourage matches in the vicinity of the projected locations.

Imagine two curves $\mathbf{B_1}(t)$ and $\mathbf{B_2}(t)$ belonging to images one and two respectively. For a location $\mathbf{p}$, the value of the curve parameter t giving the lowest distance between $\mathbf{p}$ and the curve $\mathbf{B_1}(t)$ is found, and used to calculate the corresponding location on the second curve, $\mathbf{B_2}(t)$.

Using the local gradients of the curves, the vector angle and distances between the original and projected points are preserved. This curve transform function **f** is defined as follows:

$$\mathbf{f}(\mathbf{p}) = \mathbf{B_2}(n(\mathbf{B_1}, \mathbf{p})) + \rho \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$n(\mathbf{B}, \mathbf{p}) = \arg\min_t \|\mathbf{B}(t) - \mathbf{p}\|$$

where:

$$\rho = \|\mathbf{B_1}(n(\mathbf{B_1}, \mathbf{p})) - \mathbf{p}\|$$

$$\begin{aligned} \theta = \quad & \angle\left\{(\mathbf{p} - \mathbf{B_1}(n(\mathbf{B_1}, \mathbf{p})) - \tfrac{d\mathbf{B_1}(t)}{dt}|_{t=n(\mathbf{B_1},\mathbf{p})}\right\} \\ + \quad & \angle\left\{\tfrac{d\mathbf{B_2}(t)}{dt}|_{t=n(\mathbf{B_1},\mathbf{p})} - \tfrac{d\mathbf{B_1}(t)}{dt}|_{t=n(\mathbf{B_1},\mathbf{p})}\right\} \, . \end{aligned}$$

The function $n(\mathbf{B}, \mathbf{p})$ finds the parameter t of the curve closest to the location $\mathbf{p}$. The numeric solution to this can be found in Graphic Gems [151][2]. The function **f** is now incorporated into an energy function. Given two features p and q from images one and two respectively, the function $E^{usr}(\mathbf{x})$ is proposed to encourage matches where the projected location $f(\mathbf{p})$ and feature location $\mathbf{q}$ are low:

$$E^{usr}(\mathbf{x}) = \sum_{a \in A(p)} x_a \frac{\|\mathbf{f}(\mathbf{p}) - \mathbf{q}\|^2}{r^2}$$

where r is a scalar to weight the disparity between $\mathbf{f}(\mathbf{p})$ and $\mathbf{q}$. To limit the influence of the curves to the difficult regions, the variable r is set to some distance. The value of r depends on how tightly the projected sites are to be constrained about the transformation **f**, and will vary depending on the matching difficulty of the region. By limiting the transformation to a specific area, it is possible to add multiple curves, enabling the correction of multiple difficult regions in the same image.

Encouraging Spatial Smoothness, $E^{geo}(\mathbf{x})$: Given a match between p and q, features are expected in the neighbourhood of p (p^s) to match to features in the neighbourhood of q, (q^s). The work by Berg et al. [18] from shape-matching literature describes a global geometric agreement of deformations between a set of feature matches. This idea is adapted into the energy function $E^{geo}(\mathbf{x})$,

$$E^{geo}(\mathbf{x}) = \sum_{(a,b) \in N} x_a x_b \eta(e^{\delta^2_{a.b}/\sigma^2_l} - 1) + (1 - \eta)(e^{\alpha^2_{a,b}/\sigma^2_\alpha} - 1)$$

where:

$$\delta_{(p,p^s),(q,q^s)} = \frac{|\|p - q\| - \|p^s - q^s\||}{\|p - q\| + \|p^s - q^s\|}$$

[2]As this implicitly assumes that the parameter t produces a corresponding point between the two curves, experiments were carried out to parameterise the curves by arc-length instead of t. It was found that the arc-length and t curve parameterisations did not to produce significantly different matching results.

$$\alpha_{(p,p^s),(q,q^s)} = \arccos\left(\frac{p-q}{\|p-q\|} \cdot \frac{p^s-q^s}{\|p^s-q^s\|}\right) \ .$$

The functions $\delta_{(p,p^s),(q,q^s)}$ and $\alpha_{(p,p^s),(q,q^s)}$ measure the disagreement between the distances and directions of neighbouring translational vectors respectively, with the variable $\eta \in \{0,1\}$ used to weight the importance between the two. For this work, the variance variables σ_l^2, σ_α^2 are fixed to the same values used by Torresani et al. [173]; at 4 and 4 respectively. With the variances fixed, it was found that η had the greatest effect on matching performance, values in the range $[0.4, 0.6]$ generally produce quite good results. For most scenarios, poorer results are obtained as the magnitude / angle ratio is shifted to one extreme or the other, i.e. η close to 0 or 1.

The set of neighbouring feature matches N is given by:

$$\begin{aligned} N = \{\langle (p,q),(p^s,q^s)\rangle \in |A \times A| \\ p \in N_{p^s} \vee q \in N_{q^s} \vee p^s \in N_p \vee q^s \in N_q\} \ . \end{aligned}$$

For example, given a potential match between features $p \in P'$ and $q \in P''$, $p^s \in P'$ is in the neighbourhood N_p of p, and $q^s \in P''$ is in the neighbourhood N_q of q. A feature p^s is considered to be within the neighbourhood N_p if it is within the previously defined distance r.

7.2.1 Energy Minimisation

Torresani et al. [173] use graph matching in order to find the optimal configuration for $\mathbf{x}$. However, the advantage of a globally optimum solution comes at the expense of relatively long processing times. To allow a more interactive experience, where the user is presented with updated results following the addition or modification of a Bezier curve, some observations are made that allow the fast but sub-optimal ICM [20] scheme to perform well.

Firstly, it is noted that $E^{app}(\mathbf{x})$ and $E^{usr}(\mathbf{x})$, do not depend on neighbouring feature energies, and so need only be computed once. However, $E^{geo}(\mathbf{x})$ is dependent on neighbouring feature correspondences. The number of possible feature matches to be evaluated can be reduced by considering only those within the radius r of the projection of p, $\mathbf{f(p)}$. This set of candidates is defined as $N_r(p) \subseteq A(p)$.

At the beginning of the minimisation, $E^{app}(\hat{\mathbf{x}})$ and $E^{usr}(\hat{\mathbf{x}})$ are pre-calculated, a proposed match configuration $\hat{\mathbf{x}}$ is created, and each feature p is randomly assigned to a candidate feature in $N_r(p)$. At each iteration of ICM, $E^{geo}(\hat{\mathbf{x}})$ is evaluated and used to yield $E(\hat{\mathbf{x}})$ from Equation 7.1. For each feature p, $\hat{\mathbf{x}}$ is then updated to select the entry in $N_r(p)$ with the minimum corresponding energy in $E(\hat{\mathbf{x}})$. The algorithm terminates when there are no further changes to $\hat{\mathbf{x}}$ or 10 iterations have passed. The estimated configuration is given by $\hat{\mathbf{x}}$.

7.3 Results & Discussion

Ground truth feature correspondences are very difficult to generate in real images. For the presented work, ground truth was generated for a set of wide baseline image pairs (45 deg.

Figure 7.3	MSE		No. of Feature Matches	
	Before	After	Before	After
(top)	69.71	67.82	136	199
(middle)	0.34	0.63	99	148
(bottom)	56.58	47.52	198	270

Table 7.1: Matching results before and after user interaction.

apart) using the multi-view space carving work of Starck et al. [164]. The idea is that space carving can generate very accurate, dense models of 3D objects given a large amount of views (8 in this case), and the correspondences between image pairs can then be generated from the estimated 3D meshes. The MSE between the estimated feature matches and the ground truth set then gives a measure of the match quality.

For the automatic, non-user assisted matching scheme, correspondences are found by nearest-neighbour matching of SIFT descriptors of the features. To make the comparison between user assisted and automatic matching schemes meaningful, a simple spatial consistency constraint is applied to the automatic matching strategy. Using the spatial constraint of [161], a match (p, q) also requires at least k similar matches in the neighbourhoods N_p and N_q, for these experiments, k is set to 3, and the neighbourhood radius $r = 15$. Matches are also rejected if the descriptor differences are above a threshold, which is set at 0.5. Good values of λ^{app}, λ^{usr} and λ^{geo} for Equation 7.1 were found experimentally to be 0.707, 5 and 5 respectively. Results before and after user interaction are presented in Table 7.1, and shown visually in Figures 7.3.

It is encouraging that the MSE over all the features (user and automatic) does not change much from the MSE over the automatic feature matching alone. In Figures 7.3, (top) & (bottom), the high degree of ambiguity in matching due to blur causes a higher MSE, while in Figure 7.3 (middle) the highly textured regions serve to lower the MSE. In both cases, the image conditions affect user and automatic features alike. In general, the user generated feature matches are as good as the automatic ones, despite the challenging image conditions that caused the automatic matching to fail in the first place. Additional examples of user-assisted matching are shown in Figures 7.4, 7.5 and 7.6. The results show that the algorithm is successful at dramatically increasing the number of useful matches with minimal intervention. Future work will explore the impact these new matches can have on the performance of various cinema post-production applications, such as tracking and object segmentation.

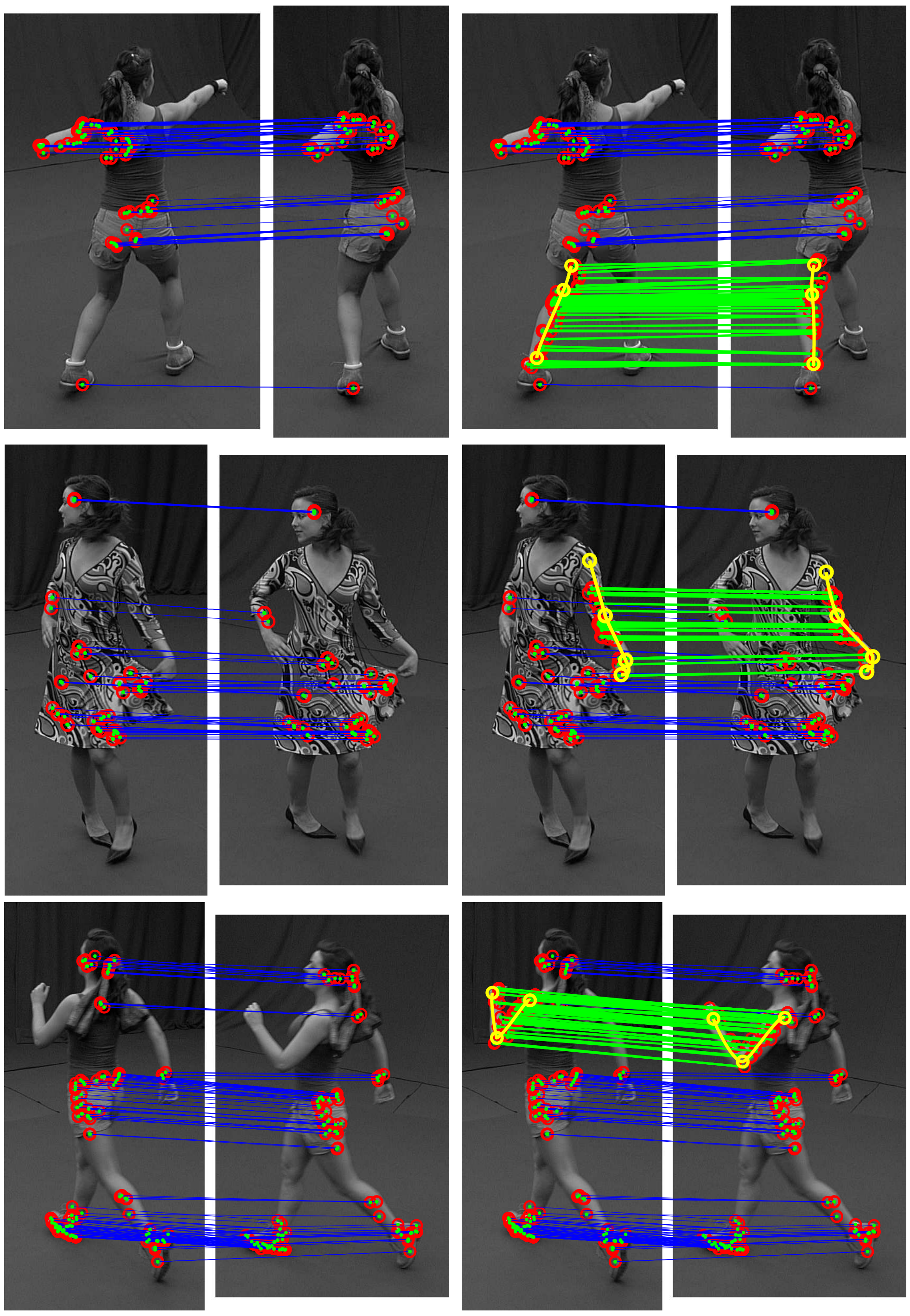


Figure 7.3: Scenes from a multi-view camera set-up. The pairs of images on the left show a sample of the strongest correspondences (blue) between the images using the scheme N. Matches are generally not found on the flat texture-less regions of the legs and arms. In the images on the right, the user supplies corresponding Bezier curves (yellow) along the lines of limbs without matches, e.g. left leg (top-right), and left arms (middle- and bottom-right). Results of the proposed guided matching are then shown in green on the right.

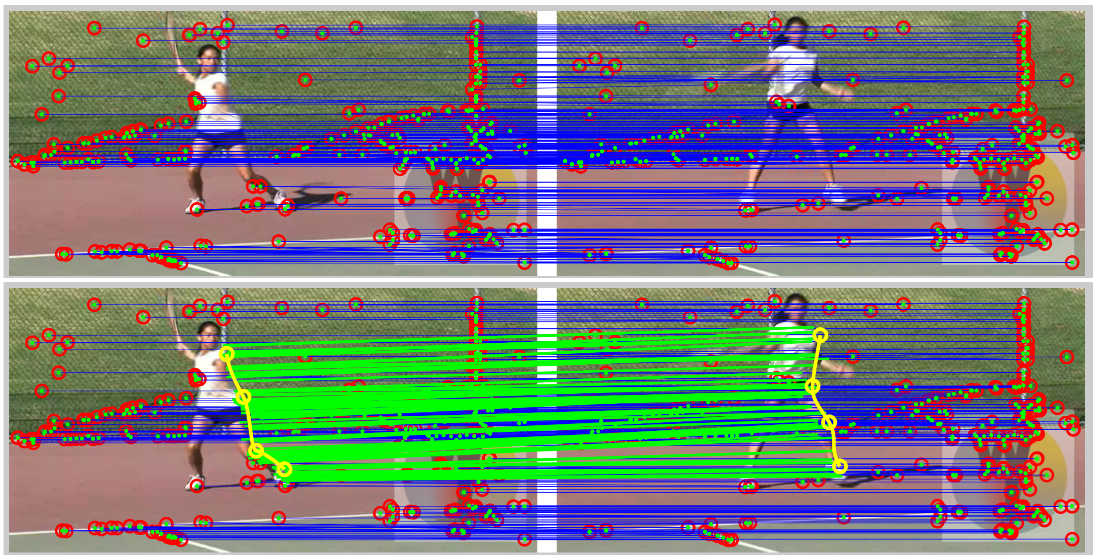


Figure 7.4: This example is interesting, although there are many correct matches in the scene, there are hardly any matches for the tennis player (top, blue). This is due to the non-rigid deformation and motion blur exhibited by the athlete. Drawing a simple set of roughly placed curves instantly generates reasonable matches for the marked side of the player.

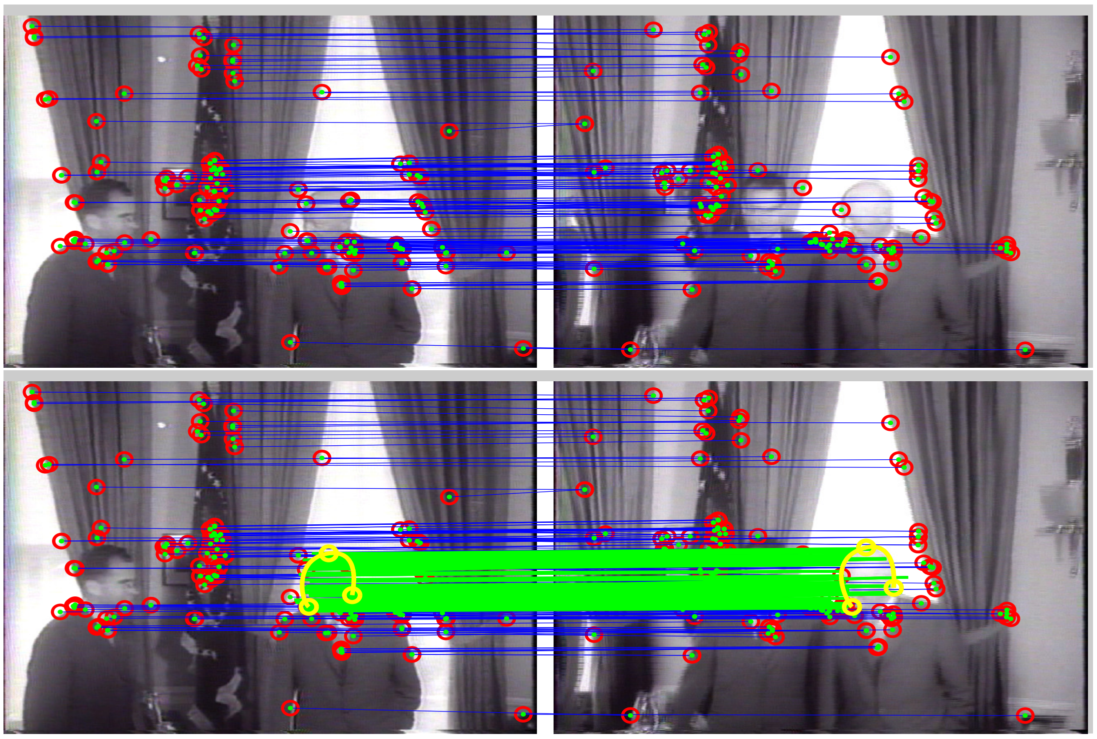


Figure 7.5: In this example severe interlacing artefacts make it very difficult to detect features. Notice that many of the originally matched features are actually incorrect (top, blue). After marking the head of the subject with a pair of curves, the algorithm has sufficient information to perform better matching.

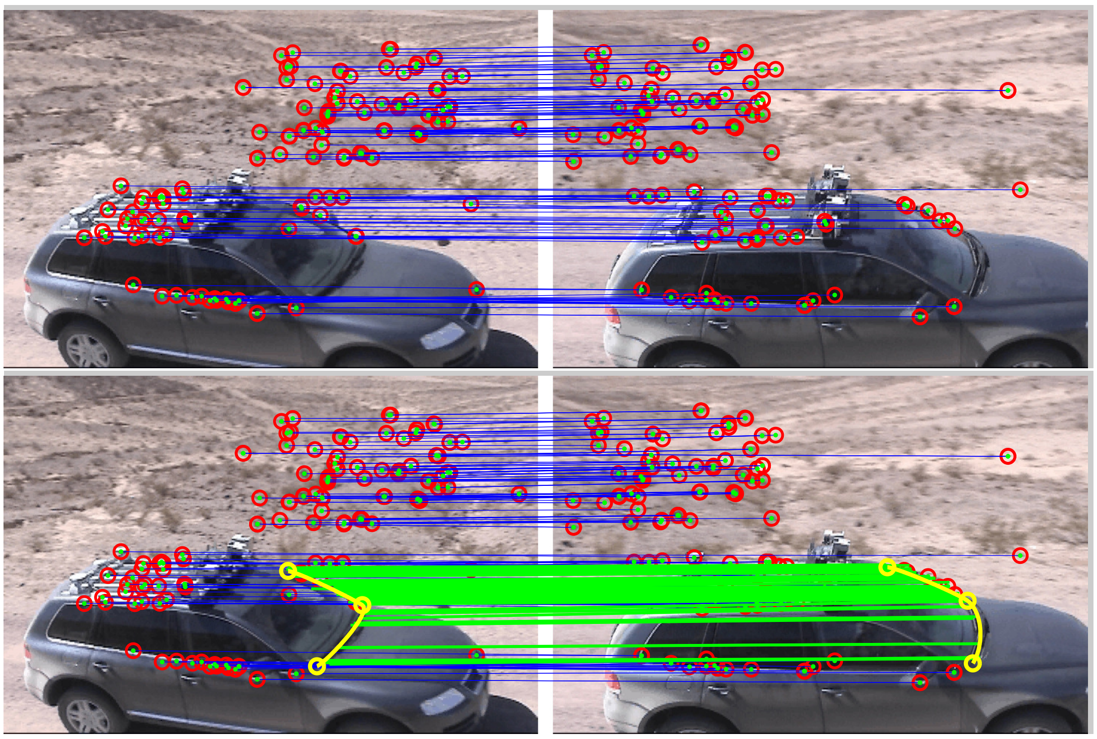


Figure 7.6: Feature matching is hindered in this video due to motion blur and reflection, particularly around the windscreen. Once a simple curve is drawn, the proposed system is able to identify many correct matches. As the car does not deform non-rigidly, the matching accuracy is higher than previous examples. (The remaining curves of the windscreen were intentionally omitted to give a clearer picture).

Chapter 8

Conclusion

This thesis investigated the use of semantically "middle level" local image features to access image content in more elaborate and interesting ways. In particular, this thesis addressed practical considerations of using local features, such as selecting appropriate detection systems and how to sensibly incorporate user interaction into the matching framework. Additionally, it was shown how feature based content analysis can be used to improve the accuracy of applications typically reserved for low level features. For example, the motion cues used to parse sports coaching video were sufficient to detect the majority of relevant athletic actions, but included a high proportion of uninteresting actions. By incorporating a simple shot clustering technique based on image content comparison, false alarm detection accuracy improved dramatically. This marks a significant departure from the traditional use of feature points in more esoteric applications such as object detection, recognition, projective geometry tasks, image registration etc.

Following on from this idea, feature points were then used in place of colour or motion features for the task of video object segmentation. This makes sense; what defines an "object" is entirely subjective, and difficult to explain at the semantically low level. Shape, colour or motion can be used to explain what an object looks like or how it moves over the short term, but have difficulty representing the underlying appearance and structure of the object. Instead, local features are designed to capture the essence of the image content. By supplying the object segmentation system with examples of what the object can look like over time, the essence of the object is recorded by the system, and used to successfully extract the object throughout the sequence with minimal user effort.

As well as demonstrating the diverse, novel use of local features in traditional applications, this thesis concentrated on many of the practical aspects of local features. This thesis attempts to answer many of the questions facing researchers wanting to use feature points in their application, such as how to select the most appropriate feature detector for a specific task, or what can be done when feature matches cannot be established between regions of known similarity. Acknowledging that feature points have their own shortcomings, and are not guaranteed to work for every situation, a semi-automatic method was presented to allow the user encourage feature matches in difficult image regions.

8.1 Future Work

The “bag of (visual) words” model of representing an image by the vector of feature occurrence frequencies is a powerful content analysis technique. As shown in Chapters 4 and 6, the bag of visual words model is not limited to large image databases, and is very useful for comparing image content in video. However, one of the practical concerns of using the bag of visual words model is how to collect enough local features representative of the video content to create a useful codebook. More features in the training set means better generalisation of the content, but longer processing times to detect and cluster the features. For future work, methods for generalising visual word vocabularies for a given detector will be investigated, with the intention of allowing the quantisation of feature descriptors without having to collect, cluster and build a large codebook first.

Feature based object segmentation demonstrated the interesting property of allowing user information to be propagated effectively forwards and backwards throughout the video, resulting in a quality object extraction with minimum effort. The two main issues to be resolved with this system are how to incorporate temporal smoothness to reduce the “popping” effect, and a faster way to match features while still including spatial consistency between the matches. Although the problem of identifying correspondences between image points is a standard problem in image processing research, there are very few matching strategies that significantly improve matching accuracy without being computationally expensive, or employing highly domain-specific knowledge. There is certainly useful information in the spatial arrangement of the feature points, but it is still unclear how the neighbourhoods of sparse feature points can be defined. A future possibility to help the matching task is to use geodesic distances to implicitly define the neighbourhoods between feature points using the actual image topology itself.

Regarding the more difficult problem of ensuring temporal smoothness, it is not clear how this can be achieved using the presented framework. The spline based extraction techniques that are currently popular in the post production industry do not exhibit the same issues with temporal smoothness, but instead have their own problems, such as getting accurate “locks” around object boundaries. Development of a hybrid system, combining the concise per-pixel cut out of the presented segmentation scheme with smoothness constraints of Bezier curves (using feature points to track image content over time) is one avenue left as future work.

8.2 Final Remarks

All of the work featured in this thesis focuses on practical scenarios using real-world images. The objective of this thesis was to demonstrate how feature points can be used in ways other than the typical image registration or object classification applications local features are usually assigned to. From the work presented in this thesis, It is easy to see how other areas that could benefit from more informative representation of image content given by local image features.

Appendix A

Precision and Recall Measures for Video Parsing

The following appendix presents a discussion for Precision and Recall with particular emphasis on their roles in measuring performance in video parsing applications. Consider the signal, $\mathbf{x} = \{x_1, \ldots, x_n\}$, of detected events generated from a video parsing application, where $x_n = 1$ indicates a detected "interesting" event at frame n, and $x_n = 0$ otherwise. Consider also the ground truth signal of interesting frames, $\mathbf{g} = [g_1, \ldots, g_n]$, where $g_n = 1$ is manually specifying frame n as being "interesting", and $g_n = 0$ otherwise. In many situations, the calculated signal, $\mathbf{x}$ can be compared directly against the ground truth signal. For example, the *mean-squared-error* (MSE), $\frac{1}{N}\|\mathbf{g} - \mathbf{x}\|^2$, or by the *sum-of-absolute-differences* (SAD), $\frac{1}{N}\sum_{n=0}^{N} |g_n - x_n|$. As the signals for this application $\mathbf{g}$ and $\mathbf{x}$ are binary vectors, the MSE and SAD functions will return the same value; the fraction of frames in the sequence incorrectly identified. However, for analysing parsing results, these measures are not very useful. Consider a typical scenario where each event lasts on average 10 frames, and the average time between each event is 90 frames. If the calculated $\mathbf{x}$ were to contain only 0's (corresponding to no actions being detected), the MSE value would be 0.1. This is a relatively low number indicating that only 10% of all frames are incorrectly labelled, despite all of the interesting actions being undetected. A different set of error measures are needed to derive anything meaningful from ground truth comparisons.

Instead of finding out the sum total of disagreement between the calculated signal $\mathbf{x}$ and the ground truth $\mathbf{g}$, both of length N frames, it is more informative to know how the errors arise. From an information retrieval (IR) point of view, any difference between signals $\mathbf{x}$ and $\mathbf{g}$ comes from either incorrectly classifying an uninteresting event as interesting (Type 1, false positive error, ϵ_α), or not detecting an interesting event (Type 2, false negative error, ϵ_β). In IR literature, Type 1 and 2 errors are usually summarised by the performance metrics, "Precision" and "Recall",

$$\begin{aligned}
N_x = & \quad \sum_{n=0}^{N} x_n \\
N_g = & \quad \sum_{n=0}^{N} g_n \\
N_c = & \quad \mathbf{x} \cdot \mathbf{g}^T \\
\text{Precision} = & \quad \frac{N_c}{N_c + \epsilon_\alpha} = \frac{N_c}{N_x} \\
\text{Recall} = & \quad \frac{N_c}{N_c + \epsilon_\beta} = \frac{N_c}{N_g}
\end{aligned}$$

where the variable N_c is the number of correctly detected frames, N_x and N_g are the numbers of frames where $\mathbf{x} = 1$ and $\mathbf{g} = 1$ respectively, the errors ϵ_α and ϵ_β are the number of "false alarm" and "missed" frames, and N is the total number of frames in the sequence. The "Precision" metric measures the fraction of correctly detected frames out of all the frames classed as actions. "Recall" is the fraction of correctly detected frames out of all frames classed as event frames in the ground truth. Intuitively, a low Precision is yielded from a large proportion of uninteresting frames in the original video appearing in the "highlights" reel, while a low Recall is observed when a high fraction of interesting frames were not recorded. When $\mathbf{x} = \mathbf{g}$, both Precision and Recall are one.

Going back to the problem scenario, where the durations of actions, and time-between-actions are 10 and 90 frames respectively. If the signal $\mathbf{x}$ has values $x_n = 0, \forall\, n \in [1, \ldots, N]$ the MSE value was 0.1. For the same scenario, the Recall value of 0 (while Precision is undefined), indicating that none of the interesting actions were correctly labelled. Conversely, if all the values of the calculated signal $\mathbf{x}$ are 1, the Recall value will be 1 and the Precision will be 0.1. The values of Precision and Recall paint a better picture of how the system is performing than MSE.

Clearly, a good segmentation will give high values for both Precision and Recall, but depending on the users preferences or application requirements, different values of Precision and Recall are tolerable. For example, in the context of the parsing application, it might be more important to return all of the interesting actions, even if that means including more uninteresting frames, i.e. desiring that Recall be higher than Precision. Or on the other hand, the user may require more concise summaries of the actions, including as little uninteresting footage as possible at the risk of omitting the occasional interesting action, accepting that Precision can be greater than Recall.

Appendix B

Automatically Assigning Actions to Detection Signals

Consider the results of the video parsing application, $\mathbf{x} = \{x_1, \ldots, x_n\}$, where $x_n = 1$ if an interesting event is detected at frame n, and $x_n = 0$ otherwise. Likewise for the ground truth signal $\mathbf{g}$. When comparing the detected signal, $\mathbf{x}$, to the ground truth signal, $\mathbf{g}$, an important factor to consider is the subjective error in the ground truth data itself. The ground truth was manually created by the user, with the user's idea of what is considered "interesting". If an action has just occurred in the video, most users will probably agree that an interesting event has taken place, but the frame boundaries of the event will vary from person to person. This ambiguity around the event frame borders increases MSE or SAD error, and lowers Precision and Recall values. Consider that actions in the ground truth and candidate actions are given by plateaus in $\mathbf{g}$ and $\mathbf{x}$ respectively. For any given action, most users will agree that a "core" set of action frames can be found within the action plateau. It makes sense then to consider that an action can be "matched" to a potential action in $\mathbf{x}$ if the overlap between the plateaus in $\mathbf{x}$ and $\mathbf{g}$ is enough to encompass a sufficient number core frames. This simple extension allows us to express system performance in *action*-based granularity, instead of frame-based granularity.

The example signals shown in Figure B.1 highlight some common problems associated with frame-based performance analysis. Notice the frames between 0 and 200, where two actions were detected for the one actual event. The detected actions probably correspond to the same event, but a temporal inconsistency has divided the actions into two, and neither action alone is correct. The low overlap of events between frames 250 and 350 is common. The detected event is composed of half an uninteresting action and half an interesting action, but probably does not capture the actual event in any useful way. The detected event between frames 400 and 500 is another common occurrence, where the detected signal $\mathbf{x}$ probably captures the true event nicely. However the small deviation from the ground truth at the onset of the event lowers the Precision dramatically. Lastly, the right-most example, between frames 500 and 800, shows the detected signal entirely encompassing three actual events. Although the events were

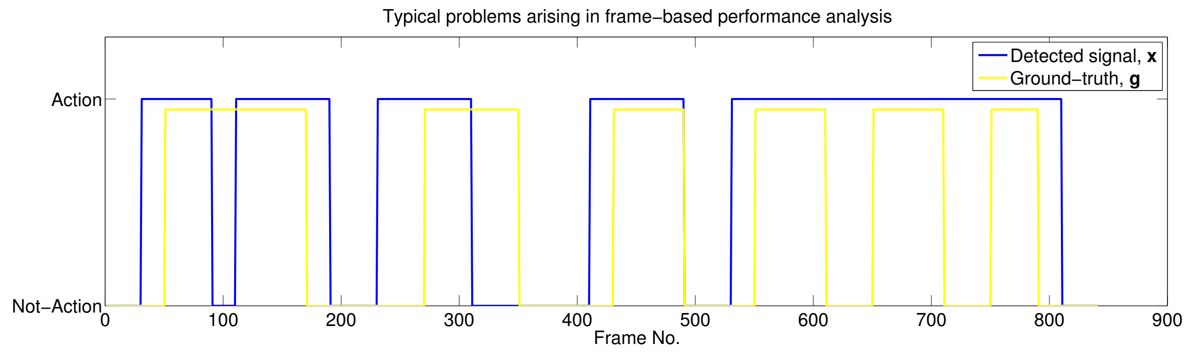


Figure B.1: Examples of problems in frame-based parsing. Although parsing errors are clearly visible between the detected **x** and **g**, the Precision and Recall for this contrived example are relatively high (0.62 and 0.85) respectively.

not distinctly separated, the Precision and Recall values will still be relatively high. In these examples it is clear to the user whether the plateaus in the detected signal correspond to the actual events shown in the ground truth. It is trivial for the user to observe either the detected plot against the ground truth or the video itself, and count the number of correctly detected events. Clearly, dealing in numbers of correct events instead of numbers of correct frames is more intuitive and indicative of the actual system performance.

Manually observing and matching-up "core" frames of plateaus between **x** and **g** takes a long time for coaching videos of reasonable (> 10 min.) length, making any type of trial and error parameter testing very difficult. It is therefore necessary to design an algorithm capable of automatically matching the detected and actual events between **x** and **g**. The idea of the proposed algorithm is to calculate the overlap between an event (plateau) in the ground truth **g**, and a candidate event in the detected signal **x**. The candidate is rejected if the size of the overlap is too small relative to the sizes of the events being compared in **g** and **x**. For the case where events are detected for the same action in **g**, as in the example of Figure B.1 (far left), it is assumed that the mapping from **g** to **x** is *injective*, that at most one action in **x** can map to an action in **g**. Similarly, at most one action in **g** can map to one action in **x**, fixing the problem example of Figure B.1 (far right). The goal now is to derive a method for detecting events based on plateaus of frames and matching them sensibly between each other.

Consider the "plateau detecting" function $\kappa(\mathbf{x}, n)$ that returns the set of frame numbers $K = \{n-u, \ldots, n, \ldots, n+v\}$ belonging to the connected action at frame n, and where $x_k = 1, \forall\ k \in K$. For example, consider if $x_n = x_{n-1} = x_{n+1} = 1$ and $\mathbf{x} = 0$ everywhere else, then $\kappa(\mathbf{x}, n) = \kappa(\mathbf{x}, n-1) = \kappa(\mathbf{x}, n+1) = \{n-1, n, n+1\}$, and $\kappa(\mathbf{x}, n+2) = \kappa(\mathbf{x}, n-2) = \emptyset$. Now consider the "step detecting" function $\gamma(\mathbf{x})$ that returns the set of frame numbers J, where $x_{j-1} = 0$ and $x_j = 1, \forall\ j \in J$. The idea is to first use the functions κ and γ to find the sets of frames for the plateaus in **x** and **g** to mark them as actions. Then, the fraction of overlap between every possible pair of actions **x** and **g** is used as the matching cost for that pair. The

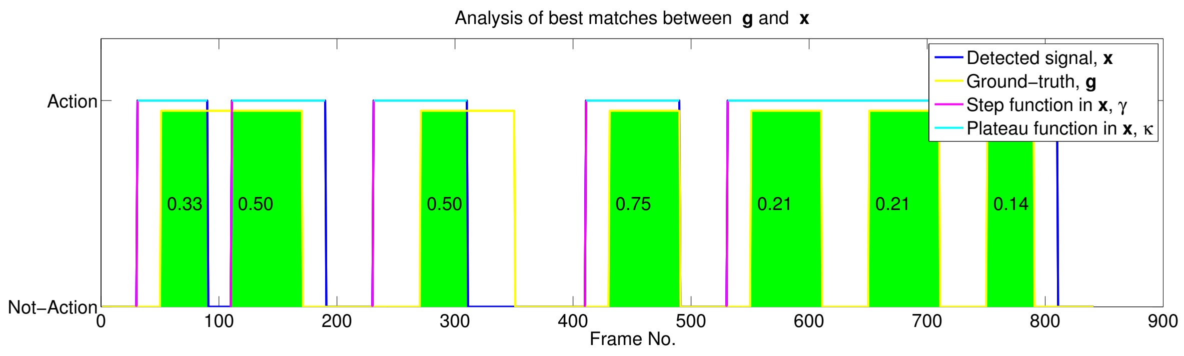


Figure B.2: Example of frame-to-action algorithm operation applied to the signals in Fig. B.1.

optimum matching arrangement for all actions is then found by solving the linear assignment problem [5]. The algorithm is formally described in Algorithm B.1.

The Algorithm B.1 is composed of two parts. The first measures the minimum fraction of overlap between an action in $\mathbf{g}$ and a candidate in $\mathbf{x}$. This fraction is then used to create the cost matrix C, where the entry in C at index (i, j) is the cost of assigning potential event j (from $\mathbf{x}$) to manually annotated event i (in $\mathbf{g}$). A perfectly overlapping pair of actions will have a cost of 0, while non-overlapping actions will have an infinite cost. Actions that overlap slightly have a cost that is inversely proportional to their overlap size, normalised by the maximum action length of the pair. This penalises actions in $\mathbf{x}$ that are both larger and smaller than the possible match in $\mathbf{g}$. A match between a pair of actions is rejected if the overlap is less than the fraction r of the maximum length of the either of the actions, and is assigned an infinite cost in C. For example, and r value of 0.75 requires that at least 75% of the frames between an action pair in $\mathbf{x}$ and $\mathbf{g}$ must overlap to be considered a match. The second part of the algorithm solves the standard *linear assignment* problem, that is, finding the match configuration of events in $\mathbf{g}$ to $\mathbf{x}$ that results in the global minimum cost, given the match cost matrix C. As the cost matrix is related to the fraction of overlap, a pair of actions are more likely to be assigned to each other if the overlap between the pair is large. In the case of multiple actions in one signal overlapping with the same action in the other signal, such as Figure B.1 (far left & far right), the action with the largest overlap is more likely to be selected as a match. This linear assignment problem is solved by the "Hungarian" algorithm [5]. Note that not all events in $\mathbf{g}$ will have a match in $\mathbf{x}$, and conversely not all events in $\mathbf{x}$ will have a match in $\mathbf{g}$. The structure of the assignment matrix A will have at most one entry of value 1 per row or column. Therefore, empty columns of A correspond to missed actions, and empty rows to false alarms, the numbers of each given by N_m and N_f respectively.

To illustrate the effects of the algorithm, Figure B.2 shows how the events in the signal $\mathbf{x}$ are detected, and the overlap between detected actions in $\mathbf{x}$ and $\mathbf{g}$ for the original plot in Fig. B.1. The detected actions in $\mathbf{x}$ are detected using a "step" detector, γ (magenta), with the

```
Input: $\mathbf{x}$, $\mathbf{g}$, $r$
Output: $N_a$, $N_d$, $N_f$, $N_m$
$S_g = \gamma(\mathbf{g})$;
$S_x = \gamma(\mathbf{x})$;
$N_a = |S_g|$;
$N_d = |S_x|$;
for $i = 1$ to $N_d$ do
    $K_i = \kappa(\mathbf{x}, S_x(i))$;
    for $j = 1$ to $N_a$ do
        $K_j = \kappa(\mathbf{g}, S_g(j))$;
        $n = |K_i \cap K_j|$;
        $O = \frac{n}{\max(|K_i|, |K_j|)}$;
        if $O < r$ then
            $C(i,j) = \infty$;
        else
            $C(i,j) = 1 - O$;
        end
    end
end
$A = \text{Hungarian}(C)$;
$N_f = \sum_{i=0}^{N_d} \delta(\sum_{j=0}^{N_a} C(i,j))$;
$N_m = \sum_{j=0}^{N_a} \delta(\sum_{i=0}^{N_d} C(i,j))$;
```

Algorithm B.1: Algorithm to detect actions from overlapping plateaus in the detected and ground truth signals $\mathbf{x}$ and $\mathbf{g}$. The parameter r is the minimum overlap threshold, with the numbers N_a, N_d, N_f and N_m corresponding to the number of actions in the ground truth, the number of detected actions, the number of false-alarms and missed actions respectively. δ is the Kronecker delta function, $\delta(x) = 1$ for $x = 0$, and $\delta(x) = 0$ otherwise.

event frames given by the plateaus connected to the step (cyan), detected by the function κ. (The same is done for $\mathbf{g}$, though this is not shown). The overlap between the detected actions is shown in green, with the minimum overlap fraction O overlaid. How O is calculated is shown in Algorithm B.1. The number of detected actions in $\mathbf{x}$ and $\mathbf{g}$ is 5 and 6 respectively. The matrix $\mathbf{O}$ of minimum overlap values between all possible event matches between $\mathbf{x}$ and $\mathbf{g}$, and the resultant cost matrix is given by:

$$\mathbf{O} = \begin{pmatrix} 0.33 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.21 & 0.21 & 0.14 \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0.5 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0.5 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0.25 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

(a) Minimum fraction of action overlap **(b)** Resulting cost matrx

Figure B.3: Results of calculation of fraction overlap and cost matrix assuming a threshold of $r = 0.5$ as given by Algorithm B.1.

where the columns of the matrices represent the events in $\mathbf{g}$, while the rows represent the events in $\mathbf{x}$. For example, the cost of assigning the detected event No. 4 in $\mathbf{x}$ to actual event No. 3 in $\mathbf{g}$ is 0.5. In this example, the linear assignment problem is simple, detected events 2, 3 & 4 in $\mathbf{x}$ are matched to events in 1, 2, & 3 in $\mathbf{g}$. The resultant detected signal with the unmatched signals in $\mathbf{x}$ removed is shown in Figure B.4. Notice that although some of the events in $\mathbf{g}$ are omitted, the new detected signal gives a more concise summary of the actual events. This is evident by the change in Precision and Recall values, i.e. from 0.62 and 0.85 using the frame based granularity, to 1 and 0.5 in the new action based granularity.

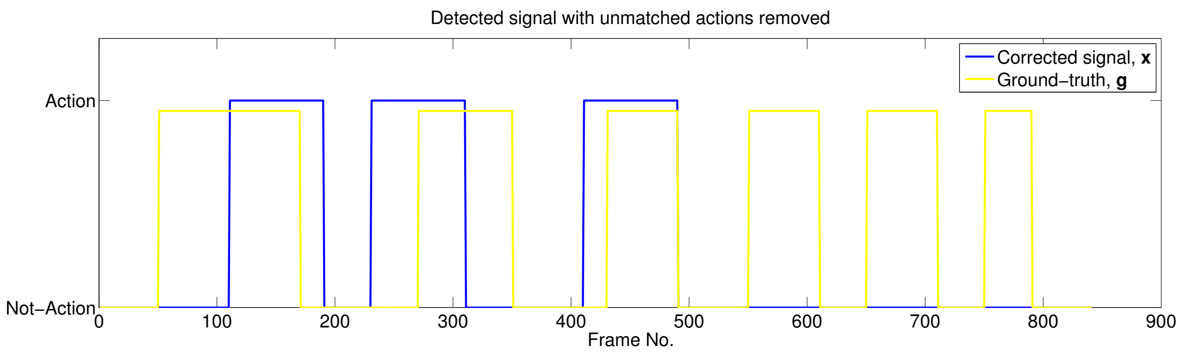


Figure B.4: Examples of removing the detected actions from the original plot in Fig. B.1 that were found not to match (or have sufficient overlap) to any corresponding action in $\mathbf{g}$.

Appendix C

The Two-Sample Kolmogorov-Smirnov Test Statistic

A large part of statistics is concerned with comparing distributions of sample populations. In Chapter 4, mean Precision and Recall values (MPR) are calculated by the movement based video parsing system using different system design choices. The goal is to compare the distributions of MPR values from the various design choices to determine if the parsing accuracy is significantly improved by including particular components. In Chapter 3, distributions of feature descriptor matches from similar and dissimilar content are compared as a measure of feature detector performance. For the distribution comparisons in Chapters 4 and 3, the shapes of the distributions can not be easily modelled by typical distribution families, such as Normal or Laplacian. This makes the comparison between distributions more difficult. Instead, the non-parametric Kolmogorov Smirnov (K.S.) test statistic is used to calculate the difference between arbitrary distributions without any prior knowledge of the distribution. To discuss the K.S. test further, the statistical evaluation of the system design choices in Chapter 4 is now presented.

Justification of Design Choices in Chapter 4 To justify the use of a particular component in the system, MPR distributions are generated by running the parsing system using thousands of combinations of system parameters. Each sample in the distribution is then added into one of two distributions, depending on whether a particular system component was omitted or included, denoted $N = 0$ and $N \neq 0$ respectively. The idea is to compare the distributions to identify any significant improvement in system performance, i.e. testing the hypothesis that the MPR values of the $N = 0$ distribution are significantly higher than those of the $N \neq 0$ distribution.

There are a number of ways to analyse the differences between a pair of distributions, for example, comparing the difference in means and variances of the two distributions (i.e. evaluating the null hypothesis $H_0 : \mu_{N\neq 0} = \mu_{N=0}$), or using a χ^2 test to compare the marginal distributions of pair-wise "greater than" comparisons between the MPR values of $N \neq 0$ and $N = 0$ pairs for each parameter combination. The first idea of comparing means and variances is complicated as prior knowledge of the shape of the distributions is needed, for example, does it belong to Normal, Students t, exponential or some other family? The second idea of the χ^2 test is useful, and can be used to test the effects of not just including or omitting one system component, but any combination of them. The χ^2 test relies on using pairs of data samples, which means that for every "negative" parameter combination, $N = 0$, a "positive", $N \neq 0$, is needed for the "greater than" comparison. This makes sense if all samples are considered to be useful, however it is obvious that most of the thousands of parameter combinations are not going to produce good results. Therefore instead of comparing all $N = 0$ samples to all $N \neq 0$ samples, only a selection of the best $N = 0$ and $N \neq 0$ samples will be used in comparison. In doing this, the pair-wise relationship between samples is lost, as a pair of samples is not guaranteed to be in both the best $N = 0$ and $N \neq 0$ sets. In the absence of any prior knowledge of the two distributions, the non-parametric, two sample K.S. test will be used to compare the distributions instead.

The statistical test comparing the pair of distributions is as follows. The null hypothesis, H_0, that there is no significant difference between the two distributions will be rejected at level α if:

$$\sqrt{\frac{nn'}{n+n'}}D_{n,n'} > K_\alpha$$

where $D_{n,n'}$ is the K.S. test statistic, K_α is the test statistic significance threshold (the "critical value" at level α), and n and n' are the number of samples in the $N \neq 0$ and $N = 0$ distributions, in this case, $n = n' = 50$. The K.S. test statistic is the maximum divergence between the cumulative distribution functions (c.d.f.'s) of the $N \neq 0$ and $N = 0$ distributions,

$$D_{n,n'} = \max\left(\mathbf{F}_n - \mathbf{F}_{n'}\right) \tag{C.1}$$

where $\mathbf{F}_n$ and $\mathbf{F}_{n'}$ are the cumulative distributions (cdf's) of the sample populations being compared. An example is shown in Figure C.1. The magnitude of the divergence effectively corresponds to the horizontal "shift in mass" between the pair of distributions, with the sign of the divergence indicating the location of the mass. For example, in Figure C.1 (right), the divergence between the c.d.f.'s is calculated as $D_{n,n'} = \max_{\mathbf{x}}(\mathbf{F}(\mathbf{x})_{N=0} - \mathbf{F}(\mathbf{x})_{N\neq 0}) = 0.46$, corresponding to the length of the red line. The positive value indicates the $N = 0$ distribution has more "mass" concentrated around lower MPR values, as can be verified by the frequency distribution plot of Figure C.1 (left). The test will be evaluated for significance at the $\alpha = 0.05$ level, with the corresponding significance threshold, K_α is given by solving the equation $P(\sqrt{\frac{nn'}{n+n'}}D_{n,n'} \leq K_\alpha) = 1 - \alpha$. In this example, with $\alpha = 0.05$ and $n = n' = 50$, the critical value, $K_\alpha = 0.1884$. As $\sqrt{\frac{5^2}{50+50}}0.46 > 0.1884$, the divergence is considered significant, and H_0 is rejected. The p value of the K.S. test is then the probability of observing a c.d.f. divergence, $D_{n,n'}$, at least as extreme as the one observed assuming the null hypothesis, H_0, is true, and is calculated as $p = P(K > \sqrt{\frac{nn'}{n+n'}}D_{n,n'})$. Marsaglia et al. [109] propose an approximation to estimating the function $P(K < d)$ for K.S. distributions. If the p value is less than α, the

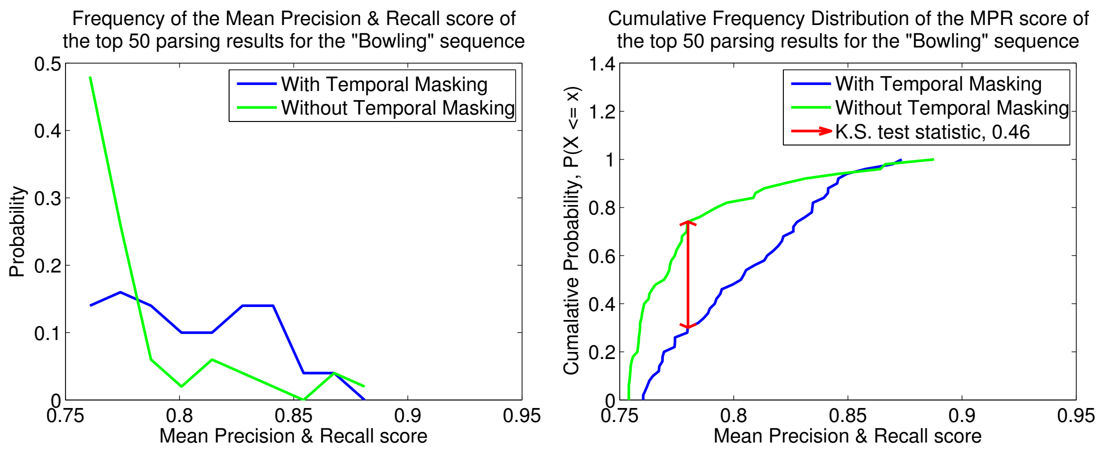


Figure C.1: Example of calculating the Kolmogorov-Smirnov (KS) test statistic for the distributions of the top 50 Mean Precision and Recall (MPR) values, with and without the temporal player mask. Left, is the probability distribution function (p.d.f.) of the MPR values. Right, the cumulative frequencies of the p.d.f.'s. The K.S. test statistic is the maximum divergence between the cumulative frequencies.

observed divergence is considered significant, and H_0 is rejected. Testing for significance with p values is equivalent, yet more intuitive than testing with critical K_α values.

Statistical Power of K.S. Test The power, $1 - \beta$ of a statistical test is the probability that the test will reject a false null hypothesis (i.e. that a Type 2 error, β will not be made). Intuitively, a low power means the test is unable to identify something as being significant when it should. This happens when the significance level α is too low[1], or more commonly when the number of samples is too low. Clearly if the number of samples is too low, a decision about whether the values of one data population are greater than the other can not be made with confidence. When the test power is high, it means very small changes to the test statistic can result in statistical significance causing H_0 to be rejected.

[1]The value α is the tolerance error in accepting a false null hypothesis, i.e. the Type 1 error. The Type 1 and Type 2 errors are inversely related, so naturally allowing for more Type 1 error will reduce the Type 2 error and increase the power of the test.

Appendix D

DVD Contents

The attached DVD contains video footage playable on standard DVD players. Presented are the highlights videos of sports coaching video, referenced from the "Visual Results" section of Chapter 4.7.1, followed by the segmentation results from the Feature based Object Segmentation framework presented in Chapter 6.4.2.

D.1 Motion Cues for Event Parsing

The results in this section relate to the on-line sports coaching video parsing application presented in Chapter 4. The videos shown here are high-light summaries as would be presented to the user. The green tick indicates a successful detection (verified by ground truth), while a red cross indicates a false alarm.

Frame based Granularity Summary videos of the motion based parsing system, compiled using the system parameters providing the highest Precision and Recall values using the frame based comparison granularity.

Action based Granularity, without False Alarm Detection Highlights videos showing the results of the parsing system, this time measuring performance using the action based granularity. The presented results are based on motion parsing only.

Action based Granularity, with False Alarm Detection Using feature based content analysis techniques, false alarms in the results of the above action based parsing are removed, resulting in a much more concise action summary.

D.2 Feature based Object Segmentation

Key-frame every 10th frame Segmentation results of the "Polo", "Quad-bike", "Oktober" & "Lady Eating Apple" sequences are calculated by selecting every 10th frame as a key-frame. In each video, the original sequence is shown, followed by the results from the presented feature based cut out algorithm. The object is then composited on a blue background using a non-binary matte derived from the algorithm results. In addition to the four "core" videos, the results for three more sequences are presented to highlight the flexibility of the system. The three clips present additional challenges, for example, image noise due to rain and large changes in object topology ("Horse Turn"), severe interlacing artefacts, poor contrast and motion blur ("Nixon"), and similar foreground / background colours and low overall image quality ("Asimo").

Information Retrieval assisted frame selection A comparison between key-frame selection methods is shown for the "Quadbike" and "Polo" sequences. Object cut out results are calculated, first by taking every 10th as a key-frame, and then using the iterative, IR based diagnostic approach presented in Chapter 6.2.1.

Bibliography

[1] Adobe Systems Incorporated. Photoshop CS4 Manual, 2008.

[2] Adobe Systems Incorporated. AfterEffects CS4, "What's New". `http://www.adobe.com/products/aftereffects/`, 2009.

[3] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. In *ACM SIGGRAPH*, pages 584–591, 2004.

[4] Y. Andreopoulos and I. Patras. Incremental refinement of image salient-point detection. *IEEE Transactions on Image Processing*, 17(9):1685–1699, 2008.

[5] H. Anton and C. Rorres. *Elementary Linear Algebra*. Wiley, 8 edition, 2000.

[6] N. Apostoloff and A. W. Fitzgibbon. Bayesian video matting using learnt image priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 407–414, 2004.

[7] N. Apostoloff and A. W. Fitzgibbon. Learning spatiotemporal t-junctions for occlusion detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 553–559, 2005.

[8] N. E. Apostoloff and A. W. Fitzgibbon. Automatic video segmentation using spatiotemporal T-junctions. In *British Machine Vision Conference*, page III: 1089, 2006.

[9] X. Bai and G. Sapiro. Distancecut: Interactive segmentation and matting of images and videos. In *IEEE International Conference on Image Processing*, volume 2, pages 249–252, October 2007.

[10] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[11] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. *ACM Transactions on Graphics*, 28(3):1–11, 2009.

[12] A. Barbu and S. Zhu. On the relationship between image and motion segmentation. In *International Workshop on Spatial Coherence for Visual Motion Analysis*, pages 51–63, 2004.

[13] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *International Joint Conferences on Artificial Intelligence*, pages 659–663, 1977.

[14] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[15] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 1000–1006, 1997.

[16] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Neural Information Processing Systems Conference*, pages 831–837, 2000.

[17] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[18] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 26–33, 2005.

[19] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, pages 237–252, 1992.

[20] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, 48:259–302, 1986.

[21] S. Beucher and F. Meyer. *The Morphological Approach of Segmentation: The Watershed Transformation*, chapter 12, pages 43–481. Mathematical Morphology in Image Processing. Marcel Dekker, New York, NY, USA, 1992.

[22] A. A. Bharath and N. Kingsbury. Phase invariant keypoint detection. In *Digital Signal Processing, International Conference on*, pages 447–450, 2007.

[23] O. Boiman and M. Irani. Detecting irregularities in images and in video. *International Journal of Computer Vision*, 74(1):17–31, 2007.

[24] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, June 1989.

[25] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *IEEE International Conference on Computer Vision*, volume 1, pages 105–112, 2001.

[26] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *IEEE International Conference on Computer Vision*, page 26, 2003.

[27] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[28] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *IEEE International Conference on Computer Vision*, pages 377–384, 1999.

[29] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[30] L. Bretzner and T. Lindeberg. Feature tracking with automatic selection of spatial scales. *Computer Vision and Image Understanding*, 71:385–392, September 1998.

[31] R. Brinkmann. *The Art and Science of Digital Compositing.* Morgan Kaufmann; 2nd edition, 2008.

[32] R. Chellappa, G. Qian, and S. Srinivasan. Structure from motion: sparse versus dense correspondence methods. In *IEEE International Conference on Image Processing*, volume 2, pages 492–499, 1999.

[33] J. G. Choi, S. W. Lee, B. J. Yun, H. S. Kang, S. H. Hong, and J. Y. Nam. Semi-automatic video object segmentation method based on user assistance and object tracking. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3214 of *Lecture Notes in Computer Science*, pages 211–218. Springer, 2004.

[34] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski. Video matting of complex scenes. *ACM Transactions on Graphics*, 21(3):243–248, July 2002.

[35] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271. IEEE Computer Society, December 2001.

[36] J. Cooper, S. Venkatesh, and L. Kitchen. The dissimilarity corner detector. In *Advanced Robotics, International Conference on*, volume 2, pages 1377–1382, June 1991.

[37] J. Cooper, S. Venkatesh, and L. Kitchen. Early jump-out corner detectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):823–828, 1993.

[38] D. Corrigan, S. Robinson, and A. Kokaram. Video Matting Using Motion Extended GrabCut. In *IET European Conference on Visual Media Production*, pages 1–9, London, UK, 2008.

[39] A. Crawford, H. Denman, F. Kelly, F. Pitie, and A. Kokaram. Gradient based dominant motion estimation with general projections for real-time video stabilisation. In *IEEE International Conference on Image Processing*, volume 5, pages 3371– 3374, October 2004.

[40] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 53–60, 2006.

[41] A. Criminisi, T. Sharp, and A. Blake. Geos: Geodesic image segmentation. In *European Conference on Computer Vision*, pages 99–112, 2008.

[42] D. Ring and F. Pitié. Feature-Assisted Sparse to Dense Motion Estimation using Geodesic Distances. In *Irish Machine Vision and Image Processing conference*, Dublin, Ireland, September 2009.

[43] Dartfish. Sport video analysis software. `http://www.dartfish.com/`.

[44] P. Delacourt, A. Kokaram, and R. Dahyot. Comparison of global motion estimators. In *Irish Signals and Systems Conference*, Cork Ireland, June 2002.

[45] C. Ding and X. He. K-means clustering via principal component analysis. In *International Conference on Machine learning*, page 29, 2004.

[46] S. Edelman, N. Intrator, and T. Poggio. Complex cells and object recognition. `http://kybele.psych.cornell.edu/~edelman/archive.html`, 1997.

[47] A. Ekin and A. M. Tekalp. Generic event detection in sports video using cinematic features. In *Computer Vision and Pattern Recognition Workshop*, volume 4, page 36, 2003.

[48] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[49] O. Faugeras, F. Lustman, and G. Toscani. Motion and structure from motion from point and line matches. In *International Conference on Computer Vision*, pages 25–34, 1987.

[50] J. Fauqueur, N. G. Kingsbury, and R. Anderson. Multiscale keypoint detection using the dual-tree complex wavelet transform. In *IEEE International Conference on Image Processing*, pages 1625–1628, 2006.

[51] M. Felsberg and G. Granlund. POI detection using channel clustering and the 2D energy tensor. In *DAGM Symposium on Pattern Recognition*, volume 3175, pages 103–110, Tübingen, Germany, 2004.

[52] R. Fielding. *The technique of special effects cinematography.* Focal Press, 4 sub. edition, 1985.

[53] 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*, 24(6):381–395, 1981.

[54] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[55] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[56] P.-E. Forssén. Maximally stable colour regions for recognition and matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, USA, June 2007.

[57] P.-E. Forssén and D. Lowe. Shape descriptors for maximally stable extremal regions. In *IEEE International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, October 2007.

[58] A. Frome and J. Malik. *Object Recognition using Locality Sensitive Hashing of Shape Contexts.* MIT Press, 2006.

[59] S. G., A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[60] C. Gallagher and A. Kokaram. Bayesian example based segmentation using a hybrid energy model. In *IEEE International Conference on Image Processing*, pages 41–44, San Antonio, Texas, USA, September 2007.

[61] X. Gao and X. Tang. Unsupervised video-shot segmentation and model-free anchorperson detection for news video story parsing. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(9):765–776, Sep 2002.

[62] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 452–472, 1990.

[63] B. Georgescu and P. Meer. Point matching under large image deformations and illumination changes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):674–688, June 2004.

[64] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[65] J. M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer. Fast anisotropic gauss filtering. *IEEE Transactions on Image Processing*, 8(12):938–943, 2003.

[66] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive alpha-matting. In *Conference on Visualization, Imaging and Image Processing*, pages 423–429, 2005.

[67] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[68] C. Gu and M.-C. Lee. Semiautomatic segmentation and tracking of semantic video objects. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 8, pages 572–584, September 1998.

[69] C. Harris. Determination of ego-motion from matched points. In *3rd Alvey Vision Confernce*, pages 189–192, September 1987.

[70] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147 – 151, 1988.

[71] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[72] A. Hitchcock. Making of “The Birds”, (Hitchcock 14 Disc Box Set). DVD, November 2005.

[73] M. Hoch and P. Litwinowicz. A semi-automatic system for edge tracking with snakes. *The Visual Computer*, 12:75–83, 1996.

[74] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[75] Imagineer Systems. Motor, ”What’s New”. `http://imagineersystems.com/products/motor/`, 2009.

[76] J. Jackman. *Bluescreen Compositing: A Practical Guide for Video & Moviemaking.* CMP Books, 2007.

[77] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[78] O. Juan and R. Keriven. Trimap segmentation for fast and user-friendly alpha matting. In *IEEE Workshop on Variational, Geometric and Level Set Methods*, volume 3752 of *Lecture Notes in Computer Science*, pages 186–197, 2005.

[79] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *European Conference on Computer Vision*, volume 1, pages 228–241, May 2004.

[80] I. Karliga and J.-N. Hwang. Analyzing human body 3-d motion of golf swing from single-camera video sequences. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 493 – 496, May 2006.

[81] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition*, pages 506–513, 2004.

[82] S. Khan and M. Shah. Object based segmentation of video using color, motion and spatial information. In *Computer Vision and Pattern Recognition*, pages 746–751, 2001.

[83] N. Kingsbury. Rotation-invariant local feature matching with complex wavelets. In *European Conference on Signal Processing*, pages 4–8, Florence, Italy, September 2006.

[84] A. Kokaram, B. Collis, and S. Robinson. Practical motion based video matting. In *IEE Conference on Visual Media Production*, pages 130– 136, London, UK, November 2005.

[85] A. Kokaram, N. Rea, R. Dahyot, M. Tekalp, P. B. P., Gros, and I. Sezan. Browsing sports video (trends in sports-related indexing and retrieval work). *IEEE Signal Processing Magazine*, 23(2):47–58, March 2006.

[86] A. C. Kokaram. *Motion Picture Restoration: Digital Algorithms for Artefact Suppression in Degraded Motion Picture Film and Video.* Springer Verlag, 1998.

[87] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26, pages 147–159, February 2004.

[88] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Obj cut. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 18–25, San Diego, CA, USA, 2005.

[89] Y. Lamdan, J. Schwartz, and H. Wolfson. Object recognition by affine invariant matching. In *Computer Vision and Pattern Recognition*, pages 335–344, June 1988.

[90] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[91] D. Lennon, N. Harte, A. Kokaram, E. Doyle, and R. Fuller. A HMM framework for motion based parsing for video from observational psychology. In *IEEE Irish Machine Vision and Image Processing Conference*, DCU, Dublin, Ireland, September 2006.

[92] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, 2008.

[93] M. Lhuillier and L. Quan. Match propogation for image-based modeling and rendering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1140–1146, 2002.

[94] B. Li and M. Sezan. Semantic sports video analysis: approaches and new applications. In *IEEE International Conference on Image Processing*, volume 1, pages 17–20, 2003.

[95] F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 524–531, 2005.

[96] H. Li, S. Lin, Y. Zhang, and K. Tao. Automatic video-based analysis of athlete action. In *International Conference on Image Analysis and Processing*, pages 205–210, 2007.

[97] W.-N. Lie, T.-C. Lin, and S.-H. Hsia. Motion-based event detection and semantic classification for baseball sport videos. In *IEEE International Conference on Multimedia and Expo*, volume 3, pages 1567–1570, June 2004.

[98] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):1573–1405, November 1998.

[99] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):1573–1405, November 1998.

[100] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: Dense correspondence across different scenes. In *European Conference on Computer Vision*, pages 28–42, Marseille, France, 2008.

[101] J. Liu and R. Hubbold. Automatic camera calibration and scene reconstruction with scale-invariant features. In *International Symposium on Visual Computing*, pages 558–568, 2006.

[102] H. Lombaert, Y. Sun, and F. Cheriet. Landmark-based non-rigid registration via graph cuts. In *International Conference on Image Analysis and Recognition*, pages 166–175, August 2007.

[103] H. Longuet. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, Sep 1981.

[104] D. G. Lowe. Local feature view clustering for 3d object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 682–688, Kauai, Hawaii, December 2001.

[105] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[106] H. Lu and Y.-P. Tan. Unsupervised clustering of dominant scenes in sports video. *Pattern Recognition Letters*, 24(15):2651–2662, 2003.

[107] F. M. Luis Alvarez. Affine morphological multiscale analysis of corners and multiple junctions. *International Journal of Computer Vision*, 25(2):95–107, November 1997.

[108] D. Marr and T. Poggio. Cooperative Computation of Stereo Disparity. *Science*, 194:283–287, October 1976.

[109] G. Marsaglia, W. W. Tsang, and J. Wang. Evaluating kolmogorov's distribution. *Journal of Statistical Software*, 8(18):1–4, November 2003.

[110] M. Marszałek and C. Schmid. Accurate object localization with shape masks. In *IEEE Conference on Computer Vision & Pattern Recognition*, June 2007.

[111] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, volume 1, pages 384–393, London, 2002.

[112] F. Meyer. An overview of morphological segmentation. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(7):1089–1118, 2001.

[113] K. Mikolajczyk, B. Leibe, and B. Schiele. Local features for object class recognition. In *IEEE International Conference on Image Processing*, pages 1792–1799, 2005.

[114] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142, 2002.

[115] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[116] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[117] 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*, 65(1/2):43–72, 2005.

[118] P. Milanfar. A model of the effect of image motion in the radon transform domain. *IEEE Transactions on Image Processing*, 8:1276–1281, September 1999.

[119] H. P. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, page 584, August 1977.

[120] H. P. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover.* PhD thesis, Stanford, CA, USA, 1980.

[121] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3d objects. *International Journal of Computer Vision*, 73(3):263–284, 2007.

[122] G. Mori. Recovering 3d human body configurations using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1052–1062, 2006.

[123] G. Mori, S. Belongie, and J. Malik. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832–1837, 2005.

[124] D. Ni, Y. Qu, X. Yang, Y. P. Chui, T.-T. Wong, S. S. Ho, and P. A. Heng. Volumetric ultrasound panorama based on 3d sift. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 52–60, 2008.

[125] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.

[126] A. Noble. *Descriptions of Image Surfaces.* PhD thesis, Department Department of Engineering Science, Oxford University, 1989.

[127] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision*, pages 490–503, 2006.

[128] D. O'Regan and A. Kokaram. Implicit spatial inference with sparse local features. In *IEEE International Conference on Image Processing*, pages 2388–2391, San Diego, CA, October 2008.

[129] M. Özuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[130] H. Pan, P. van Beek, and M. I. Sezan. Detection of slow-motion replay segments in sports video for highlights generation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1649–1652, 2001.

[131] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[132] F. Pitié, S.-A. Berrani, R. Dahyot, and A. Kokaram. Off-line multiple object tracking using candidate selection and the viterbi algorithm. In *IEEE International Conference on Image Processing*, volume 3, pages III– 109–12, Genoa, September 2005.

[133] F. Pla, P. Ribeiro, J. Santos-Victor, and A. Bernardino. Extracting motion features for visual human activity representation. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 537–544, 2005.

[134] A. Pooransingh, C. Radix, and A. Kokaram. The path assigned mean shift algorithm: A new fast mean shift implementation for colour image segmentation. In *IEEE International Conference on Image Processing*, pages 597–600, 2008.

[135] R. B. Potts. Some generalized order-disorder transformations. *Cambridge Philos. Soc. Math. Proc.*, 48:106–109, 1952.

[136] B. L. Price, B. S. Morse, and S. Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *International Conference on Computer Vision*, Kyoto, Japan, September 2009.

[137] P. Rajan and J. Davidson. Evaluation of corner detection algorithms. In *Twenty-First Southeastern Symposium on System Theory*, pages 29–33, March 1989.

[138] N. Rea, R. Dahyot, and A. Kokaram. Semantic event detection in sports through motion understanding. In *International Conference on Image and Video Retrieval*, pages 88–97, 2004.

[139] A. Rebai, A. Joly, and N. Boujemaa. Constant tangential angle elected interest points. In *ACM International workshop on Multimedia Information Retrieval*, pages 203–212, 2006.

[140] C. Rhemann, C. Rother, A. Rav-Acha, and T. Sharp. High resolution matting via interactive trimap segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[141] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. A perceptually motivated online benchmark for image matting. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2009.

[142] P. Ribeiro and J. Santos-Victor. Human activity recognition from video: modeling, feature selection and classification architecture. In *Human Activity Recognition and Modelling, Workshop on*, pages 61–70, 2005.

[143] P. C. Ribeiro and J. Santos-Victor. Human activity recognition from video: modeling, feature human activity recognition from video: modeling, feature selection and classification architecture. In *British Machine Vision Conference Workshop*, pages 61–70, October 2005.

[144] D. Robinson and P. Milanfar. Fast local and global projection- based methods for affine motion estimation. *Journal of Mathematical Imaging and Vision*, 18:35–54, January 2003.

[145] M. Roh, W. Christmas, J. Kittler, and S. Lee. Robust player gesture spotting and recognition in low-resolution sports video. In *European Conference on Computer Vision*, volume 4, pages 347–358, 2006.

[146] M. Roh, W. Christmas, J. Kittler, and S. Lee. Gesture spotting for low-resolution sports video annotation. *Pattern Recognition*, 41(3):1124–1137, March 2008.

[147] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (to appear)*, 2009.

[148] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[149] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[150] C. Schmid and A. Zisserman. The geometry and matching of lines and curves over multiple views. *International Journal of Computer Vision*, 40(3):199–233, 2000.

[151] P. J. Schneider. An algorithm for automatically fitting digitized curves. In *In Graphics Gems*, pages 612–626. Academic Press, 1990.

[152] P. Schugerl, R. Sorschag, W. Bailer, and G. Thallinger. Object re-detection using sift and mpeg-7 color descriptors. In *Multimedia Content Analysis and Mining*, pages 305–314, 2007.

[153] E. Seemann, B. Leibe, K. Mikolajczyk, and B. Schiele. An evaluation of local shape-based features for pedestrian detection. In *British Machine Vision Conference*, pages 11–20, Oxford, UK, 2005.

[154] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.

[155] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[156] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[157] E. Sifakis, I. Grinias, and G. Tziritas. Video segmentation using fast marching and region growing algorithms. *EURASIP Journal on Advances in Signal Processing*, 2002(4):379–388, 2002.

[158] I. Simon and S. Seitz. A probabilistic model for object recognition, segmentation, and non-rigid correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.

[159] S. N. Sinha, J. michael Frahm, M. Pollefeys, and Y. Genc. Gpu-based video feature tracking and matching. Technical report, Workshop on Edge Computing Using New Commodity Architectures, 2006.

[160] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[161] J. Sivic and A. Zisserman. Video Google: Efficient visual search of videos. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 2006.

[162] J. Sivic and A. Zisserman. Efficient visual search for objects in videos. *Proceedings of the IEEE*, 96(4):548–566, 2008.

[163] P. Soille. *Morphological Image Analysis: Principles and Applications.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[164] J. Starck and A. Hilton. Surface capture for performance based animation. *IEEE Computer Graphics and Applications*, 27:21–31, 2007.

[165] M. B. Stegmann, B. K. Ersbøll, and R. Larsen. FAME – a flexible appearance modelling environment. *IEEE Transactions on Medical Imaging*, 22(10):1319–1331, 2003.

[166] J. Stoettinger, A. Hanbury, N. Sebe, and T. Gevers. Do colour interest points improve image retrieval? In *IEEE International Conference on Image Processing*, volume 1, pages 169–172, October 2007.

[167] J. Sun, S. B. Kang, Z.-B. Xu, X. Tang, and H.-Y. Shum. Flash cut: Foreground extraction with flash and no-flash image pairs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[168] R. Szeliski. Image alignment and stitching: a tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2(1):1–104, 2006.

[169] The Foundry. Nuke manual. `http://www.thefoundry.co.uk/`, 2008.

[170] P. J. Toivanen. New geodesic distance transforms for gray-scale images. *Pattern Recognition Letters*, 17(5):437–450, 1996.

[171] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[172] P. H. S. Torr. Geometric motion segmentation and model selection. *Philosophical Transactions of the Royal Society of London A*, 356:1321–1340, 1998.

[173] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *European Conference on Computer Vision*, pages 596–609, 2008.

[174] B. Triggs. Detecting keypoints with stable position, orientation and scale under illumination changes. In *European Conference on Computer Vision*, volume 4, pages 100–113, May 2004.

[175] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[176] H. Wang and M. Brady. Real-time corner detection algorithm for motion estimation. *Image and Vision Computing*, 13(9):695–703, 1995.

[177] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Transactions on Graphics*, 24(3):585–594, 2005.

[178] J. Wang, R. Cipolla, and H. Zha. Vision-based global localization using a visual vocabulary. In *IEEE International Conference on Robotics and Automation*, pages 4230–4235, April 2005.

[179] J. Wang and M. F. Cohen. Image and video matting: a survey. *Foundations and Trends in Computer Graphics and Vision*, 3(2):97–175, 2007.

[180] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

[181] P. R. White, B. Collis, S. Robinson, and A. Kokaram. Inference matting. In *IEE European Conference on Visual Media Production*, pages 168–172, London, UK, November 2005.

[182] J. Wills, S. Agarwal, and S. Belongie. A feature-based approach for dense segmentation and estimation of large disparity motion. *International Journal of Computer Vision*, 68(2):125–143, 2006.

[183] S. A. Winder and M. Brown. Learning local image descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[184] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[185] Z. Xiong, X. S. Zhou, Q. Tian, Y. Rui, and H. Ts. Semantic retrieval of video - review of research on video retrieval in meetings, movies and broadcast news, and sports. *IEEE Signal Processing Magazine*, 23(2):18–27, 2006.

[186] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis. Improved fast gauss transform and efficient kernel density estimation. *IEEE International Conference on Computer Vision*, 1:464, 2003.

[187] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20(1):68–86, 1971.

[188] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, 2007.

[189] H. Zhou, Y. Yuan, and C. Shi. Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, August 2008.