

Computer Vision to See People:
a basis for enhanced human computer interaction.

Gareth Loy

A thesis submitted for the degree of
Doctor of Philosophy
at the Australian National University



Robotics Systems Laboratory
Department of Systems Engineering
Research School of Information Sciences and Engineering
Australian National University

January 2003



Statement of Originality

These doctoral studies were conducted under the supervision of Professor Alexander Zelinsky. The work submitted in this thesis is a result of original research carried out by myself, except where duly acknowledged, while enrolled as a PhD student in the Department of Systems Engineering at the Australian National University. It has not been submitted for any other degree or award.

A handwritten signature in cursive script, reading "Gareth Loy". The signature is written in black ink and is positioned to the right of the printed name.

Gareth Loy

Acknowledgements

Firstly I would like to thank my supervisor Professor Alexander Zelinsky both for his vision, insight and motivation, and the tremendous opportunities which he provided to me during my PhD studies. Thankyou also to the other academics in the department, in particular Professor Richard Hartley, Professor John Moore, and Dr David Austin whose guidance, support and insight have been invaluable.

To my fellow students in Systems Engineering with whom I've shared the highs and lows of post-graduate study, and who made the department such a great place to work, Roland Goecke, Rochelle O'Hagan, Dr Jochen Heinzman, Dr Jeremy Thorne, Wayne Dunstan, Dr Tanya Conroy, Dr Louis Shue, Dr Llew Mason, Leanne Matuszyk, Grant Grubb, Dr Simon Thompson, Dr Chris Gaskett, and Matthew Smith, and special thanks to Luke Fletcher and Nicholas Apostollof with whom I have had the pleasure to work with over the last year. Thankyou also to the non-academic staff, James Ashton, Rosemary Shepard, Jenny Watkins, Marita Rendina, and Karen Montefiore for all the assistance they have provided me over the last few years.

During my PhD I was fortunate enough to spend several months at the University of Western Australia in 2000, and The Humanoid Interaction Laboratory at AIST, Tsukuba, Japan in 2002. Thankyou to Dr Eunjung Holden and Professor Robyn Owens for hosting me at the University of Western Australia and for the exciting time we had working together, and thankyou to all my friends in Perth who made my time there so enjoyable. Thankyou to Dr Gordon Cheng, Professor Yasuo Kuniyoshi and Yoko Sato for hosting me in Tsukuba and for making my stay such an enjoyable experience, and to my friends at Ninomiya House who made Tsukuba a home away from home.

To the many other fabulous people who have touched my life over the few last years, in particular Tessa Du, David Moyle, Louise DeSantis, Arianne Lowe, Anh Nguyen, Dr Daniel Ayers, Marcus Uzubalis, Dr David & Yaeli Liebowitz,

Damien Halliday, Emily Nicholson, Rosemary Driscoll, Shoko Okada, Catherine Moyle, Olivia Grey-Rodgers, Natalie Martino, Edwina Hopkins, Justine Lamond, Lars Petersson, Dirk Platzen and Annette Kimber, thankyou for your friendship, vitality and for the good times we shared. Special thanks to Jessica Lye and Nina Amini for so many things, but especially for being truly outstanding friends.

Lastly, and most importantly I would like to thank my family, Rick, Winifred, Adèle and more recently Scott, who have always been there for me providing love, guidance, encouragement and support throughout my life.

Gareth Loy

Abstract

Vision is the primary sense through which people perceive the world, and the importance of visual information during our interactions with people is well known. Vision can also play a key role in our interaction with machines, and a machine that can see people is more able to interact with us in an informed manner. This thesis describes work towards a computer vision system to enable a computer to see people's faces, and hence provide a basis for more meaningful and natural interaction between humans and computers.

The human face possesses a number of visual qualities suitable for detecting faces in images. Radial symmetry is particularly useful for detecting facial features. We present new transform, the Fast Radial Symmetry Transform (FRST), that allows efficient computation of local radial symmetry in realtime. Both as a facial feature detector and as a generic region of interest detector the FRST is seen to offer equal or superior performance to existing techniques at a comparatively low computational cost.

However, no single cue can perform reliably in all situations. The key to an efficient and robust vision system for tracking faces or other targets is to intelligently combine information from a number of different cues, whilst effectively managing the available computational resources. We develop a system that adaptively allocates computational resources over multiple cues to robustly track a target in 3D.

After locating and tracking a face in an image sequence, we look at the problem of detecting facial features and verifying the presence of a face. We present an automatic face registration system designed to automatically initialise features for a head tracker. We also explore the problem of tracking the facial features. This involves tracking both rigid and deformable features to determine the 3D head pose, and describe the locations of facial features relative to the head. The 3D head pose is tracked using predominantly rigid facial features, and deformable

features are then tracked relative to the head. A new form of templates was introduced to facilitate tracking deformable features. These are used in two case studies. The first is a monocular lip tracker, and the second is a stereo lip tracking system that tracks the mouth shape in 3D.

The face localisation, feature detection and tracking solutions presented in this thesis could potentially be integrated to form an all-inclusive vision system allowing a computer or robot to really see a person's face.

Publications Resulting from this Thesis

Journal Publication

- Gareth Loy and Alexander Zelinsky. Fast Radial Symmetry for Detecting Points of Interest. *IEEE Trans on Pattern Analysis and Machine Intelligence*, Vol. 25, No 8, pp. 959-973, August 2003.

Conference Publications

- Gareth Loy and Alexander Zelinsky. A Fast Radial Symmetry Transform for Detecting Points of Interest. *Proc of European Conference on Computer Vision (ECCV2002)*. Copenhagen, May 2002.
- Gareth Loy, Luke Fletcher, Nicholas Apostoloff and Alexander Zelinsky. An Adaptive Fusion Architecture for Target Tracking. *Proceedings of Fifth International Conference on Face and Gesture Recognition (FGR2002)*, Washington DC, May 2002.
- Gareth Loy, Roland Goecke, Sebastian Rougeaux and Alexander Zelinsky. Stereo 3D Lip Tracking. *Proceedings of Sixth International Conference on Control, Automation, Robotics and Computer Vision (ICARCV2000)*, Singapore, December 2000.
- Eunjung Holden, Gareth Loy and Robyn Owens. Accommodating for 3D head movement in visual lipreading. *Proceedings of International Conference on Signal and Image Processing (SIP)*, pp. 166-171, 2000.
- Gareth Loy, Eunjung Holden and Robyn Owens. A 3D Head Tracker for an Automatic Lipreading System. *Proceedings of Australian Conference on*

Robotics and Automation (ACRA2000), Melbourne Australia, August 2000.

Provisional Patent

- Provisional Patent #PS1405, Method for Automatic Detection of Facial Features, Gareth Loy, Seeing Machines Pty Ltd, 27 March 2002.

Contents

Statement of Originality	iii
Acknowledgements	v
Abstract	vii
Publications Resulting from this Thesis	ix
1 Introduction	1
1.1 Principal Objectives	3
1.2 Key Contributions	5
1.3 Outline of Thesis	5
1.3.1 Related Work	6
1.3.2 A Fast Radial Symmetry Transform	6
1.3.3 Face Localisation	6
1.3.4 Face Registration	7
1.3.5 Face Tracking	7
1.3.6 Conclusion	8
1.4 Chapter Summary	8
2 Related Work	9

2.1	Cues for Person Tracking	9
2.1.1	The Human Face	9
2.1.2	Skin Detection	13
2.1.3	Depth Maps	19
2.1.4	Motion	26
2.1.5	Radial Symmetry Operators	30
2.2	Face Localisation	37
2.3	Face Registration	43
2.4	Face Tracking	47
2.4.1	Tracking Rigid Facial Features	48
2.4.2	Tracking Deformable Facial Features	55
2.5	Summary	61
3	Fast Radial Symmetry Detection	63
3.1	Definition of the Transform	64
3.2	Choosing the Parameters	68
3.2.1	Set of Radii N	68
3.2.2	Gaussian Kernels \mathbf{A}_n	69
3.2.3	Radial-strictness Parameter α	71
3.2.4	Normalizing Factor k_n	71
3.3	Refining the Transform	74
3.3.1	Ignoring Small Gradients	75
3.3.2	Dark & Bright Symmetry	75
3.3.3	Choosing a Constant \mathbf{A}_n	77
3.4	A General Set of Parameters	77

CONTENTS	xiii
3.5 Performance Evaluation	78
3.5.1 Performance of the FRST	78
3.5.2 Comparison with Existing Transforms	83
3.6 Summary	90
4 Face Localisation	91
4.1 A Bayesian Approach to Target Localisation	92
4.1.1 Markov Localisation	92
4.1.2 Markov Localisation with a Particle Filter	94
4.2 System Design	95
4.2.1 Particle Filter	96
4.2.2 Cue Processor	98
4.3 Localising and Tracking a Head in a Complex Environment	102
4.3.1 Implementation	102
4.3.2 Preprocessing	102
4.3.3 Hypothesis Testing	105
4.3.4 Performance	108
4.4 Tracking Multiple Targets	110
4.4.1 Multiple Particle Filters	111
4.4.2 Experimental Setup	111
4.4.3 Results	113
4.5 Summary	118
5 Face Registration	121
5.1 Automating the Detection of Features	122

5.2	Target Specification	123
5.3	Description of the System	126
5.3.1	Detecting Blink-like Motion	126
5.3.2	Extraction of Face Candidate Region	129
5.3.3	Enhancement of Features	131
5.3.4	Classifying Facial Features	135
5.3.5	Verify Face Topology	145
5.3.6	Checking Similarity of Feature Pairs	145
5.4	Performance of System	146
5.4.1	Implementation	146
5.4.2	Detection Performance	147
5.4.3	Seeing Machines System	149
5.5	Summary	150
6	Face Tracking	151
6.1	Adaptable Templates	152
6.2	Monocular Lip Tracking	156
6.2.1	Monocular 3D Head Tracker	157
6.2.2	Mouth Detection and Correction for Pose	166
6.2.3	Experimentation	167
6.2.4	Section Review	172
6.3	Stereo Lip Tracking	172
6.3.1	Stereo Vision System	173
6.3.2	Head Tracking	173
6.3.3	Lip Tracking	174

CONTENTS	xv
6.3.4 Experimentation	180
6.3.5 Section Review	185
6.4 Summary	186
7 Conclusion	187
7.1 Summary	187
7.2 Achievements	189
7.2.1 Fast Detection of Radial Symmetry	189
7.2.2 An Adaptive Fusion Architecture for Target Tracking . . .	189
7.2.3 Facial Feature Detection	190
7.2.4 3D Deformable Facial Feature Tracking	190
7.3 Further Work	191
A Contents of CD-ROM	193
B Derivation of the Optical Flow Constraint Equation	195

List of Figures

1.1	Humans communicating	2
1.2	Enabling a computer to “see” a face	4
2.1	Facial qualities suitable for detection by a computer	10
2.2	Average face	12
2.3	Constructing a skin colour model	15
2.4	Detecting skin	16
2.5	A stereo image pair and associated depth map	20
2.6	Pinhole camera model and stereo camera configurations	21
2.7	Laplacian of Gaussian	24
2.8	Difference of Gaussian	24
2.9	Example of a 3×3 neighbourhood centred on a point p	25
2.10	Result of Zabih and Woodfill’s rank transform with radius 1	25
2.11	Examples of different motion cues	27
2.12	Modelling fixation tendencies	31
2.13	Examples of Reisfeld <i>et al.</i> ’s Generalised Symmetry Transform	31
2.14	Gradient orientation masks used by Lin and Lin	32
2.15	Inverted annular template as used by Sela and Levine	34
2.16	The spoke filter template proposed by Minor and Sklansky	35

2.17	Example of Di Gesù and Valenti's Discrete Symmetry Transform	35
2.18	Example of Kovese's symmetry from phase	36
2.19	Evolution of particles over a single time-step	38
2.20	Cues operating in Triesch and von der Malsburg's system	41
2.21	Triesch and von der Malsburg's system in operation	42
2.22	Face registration	44
2.23	The kernel used by Yow and Cipolla (1997)	45
2.24	3D pose of a head and head reference frame	49
2.25	3D reconstruction from stereo images.	51
2.26	A right angle triangle from the i^{th} camera in Figure 2.25	52
2.27	Example of Matsumoto and Zelinsky's head tracking system	56
2.28	The appearance of a subject's mouth can vary greatly	57
2.29	Revéret and Benoît's lip model	60
3.1	Steps involved in computing the FRST	65
3.2	The locations of affected pixels	66
3.3	Effect of varying radii at which the FRST is computed	70
3.4	The contribution of a single gradient element	70
3.5	Effect of varying α	72
3.6	Some example images from the test set	73
3.7	Mean and standard deviation of the maximum of \mathbf{O}_n	74
3.8	The effect of different values of β on S	76
3.9	Examples of dark and bright symmetries	76
3.10	256×256 lena image	78
3.11	Results of applying the FRST to the 256×256 lena image	80

3.12	The FRST applied to face and other images	81
3.13	The FRST being calculated online in realtime	82
3.14	Comparison of performance on an outdoor image	87
3.15	Comparison of performance on the standard lena image	88
3.16	Comparison of performance on an image of a face in half shadow .	89
4.1	The four steps of the particle filter.	95
4.2	System architecture.	95
4.3	Particle filter tracking a head in (x, y, z, θ) state space	97
4.4	Example of particle population evolving over time	98
4.5	Sensing process	103
4.6	Preprocessing a colour stereo image pair	104
4.7	Generic head target and associated search regions	106
4.8	Several frames in tracking sequence	108
4.9	Frame in tracking sequence	109
4.10	Cue utility and associated processing delay	110
4.11	Two particle filters tracking separate targets	112
4.12	Preprocessing results from single camera	114
4.13	Some snapshots of the system running	115
4.14	Some snapshots of the system running	116
4.15	Some snapshots of the system running	117
5.1	Average face and facial dimensions	124
5.2	Average face showing placement of the mouth and nose	125
5.3	Structure of face-finding algorithm.	126

5.4	Detection of blink-like motion.	128
5.5	Process for extracting potential face region from image buffer. . .	130
5.6	Face region defined in terms of the interpupillary distance	131
5.7	Process for enhancing features in face region.	132
5.8	Images associated with the enhancement process.	133
5.9	Regions S_i used by L_i for enhancing facial features	134
5.10	Procedure for locating facial features.	135
5.11	Process for locating feature rows using integral projection	136
5.12	Process for locating feature columns within each feature row . . .	136
5.13	Closeup view of a human eye.	138
5.14	Process for locating eyes.	138
5.15	Region used by local comparison operator to highlight sclera . . .	139
5.16	Process for locating mouth corner.	140
5.17	Local comparison operator regions for enhancing the mouth . . .	141
5.18	Locating nostrils	142
5.19	Elimination of non-plausible nostril pairs	143
5.20	Eyebrow detection.	144
5.21	The similarity of symmetrically opposite features is verified. . . .	146
5.22	Snapshots of a sequence	147
5.23	Results of the system on a range of subjects	148
5.24	Results from the Seeing Machines implementation	149
6.1	Template matching	153
6.2	Template matching a deformable feature using NCC	155
6.3	3D pose of a head and head reference frame	157

6.4	The process the head tracker steps through each frame.	158
6.5	Pinhole camera model.	161
6.6	Face image showing projected feature locations and search regions	162
6.7	Searching procedure using initial sparse search	163
6.8	Search lines for locating the top and bottom mouth edges.	167
6.9	Projection of mouth points from image plane to face plane.	168
6.10	Some snap shots of the system in operation	170
6.11	Results of the head and mouth tracking system	171
6.12	Overview of the system.	173
6.13	The stereo camera arrangement.	174
6.14	Primary tracking points	175
6.15	Lip tracking system.	175
6.16	Search lines placement	178
6.17	Contour tracking templates	179
6.18	The system in operation	181
6.19	3D error in primary feature locations.	182
6.20	Absolute error in x -direction in primary feature locations.	183
6.21	Absolute error in y -direction in primary feature locations.	183
6.22	Absolute error in z -direction in primary feature locations.	184

List of Tables

2.1	Face Dimensions of British Adults	11
2.2	Head Dimensions Bounding Populations	11
3.1	Parameter Settings used for Experimentation	69
3.2	Parameter Settings used for Experimentation	78
3.3	Estimated Computation Required for Different Transforms	84
3.4	Computational Order of Different Transforms	86
5.1	Estimated Computations per Frame	147
6.1	Estimated Computations per Frame	169
6.2	Mean absolute error in primary tracking points.	182
6.3	Estimated Computations per Frame	184

Chapter 1

Introduction

Interpersonal communication is a central part of people's lives. The purposes of communicating with other people are many and varied. We commonly communicate who we are, what we are doing, or how we are feeling, and instruct others how to do things, or what we would like them to do. People communicate effortlessly using language, tone of voice, gestures, posture and facial expressions. A significant proportion of this communication is non-verbal. Figure 1.1 shows people communicating in different circumstances — just by observing the people in these pictures we can tell quite a lot about their situations, and begin to guess what it is they are communicating.

There is some debate amongst experts as to exactly how much interpersonal communication is non-verbal. Birdwhistell (1970) estimates 65 percent of the information transferred in a normal two person conversation is non-verbal, whereas Mehrabian (1972) postulates it to be as high as 93 percent. The precise value is of little consequence, the point is that non-verbal — as well as verbal — communication plays a crucial role in the interaction between people.

Compared to the way people interact with each other our interaction with computers (and robots) is much more restricted. Traditionally people have interacted with computers using a keyboard and mouse or other pointing device, and whilst these are well suited for most standard computer tasks it limits computers to these “standard” tasks. By standard tasks we mean tasks that computers are traditionally considered as being good at, such as word processing, database management, programming, browsing the internet, analyzing data, and performing numerical computations.



Figure 1.1: Humans communicating.

Enhancing the interaction between humans and computers offers many new possibilities. Ideally we would like to be able to interact with a computer in the same way we do with another person. This would open the door to many new and useful applications. Potential tasks could include:

- entertainment, interfaces for games, facial animation,
- improved teleconferencing,
- monitoring human performance,
- classroom teaching,
- caring for the elderly, or the disabled,
- smart cars, smart devices,
- smart security surveillance, and
- making manual or automated tasks easier.

Computer vision allows computers to “see”. Having a computer that can see a person is a significant step closer to a machine that we can interact with. The research in this thesis has focused on helping a computer to see a person, in particular the face.

1.1 Principal Objectives

The goal of this research is to work towards a computer vision system that enables a computer to see people's faces, and hence provide a basis for more meaningful and natural interaction between humans and computers. What do we mean when we say we want a computer to be able to "see" faces? We want the computer to be able to locate and track humans in image sequences, preferably in real-time, and with robustness to different people's appearances and the operational environment.

There are a number of aspects to this problem. Firstly it is necessary to know where a person is in a scene, in particular the location of their head. Once the approximate location of the head is known the facial features can be detected, and once these features have been found they can be used to track the pose of the head and the relative motion of deformable features such as the mouth and eyebrows. This thesis aims to present solutions to these human tracking issues that could potentially form an all-inclusive vision system to allow a computer or robot to see a person's face.

Figure 1.2 shows how the problem of enabling a computer to see faces can be broken down into face localisation, face registration, and face tracking. It also shows how these different stages of the process relate directly to human computer interaction applications such as human motion capture, face recognition, lip reading and expression recognition.

The first, and perhaps the most challenging problem to be dealt with, is face localisation. Consider the situation where a person is moving around a room, the lighting conditions are variable, sometimes there are objects occluding the person's face, there may even be more than one subject to be tracked, and the camera is not assumed to be stationary. The face localisation module must robustly locate the approximate location of the face and track it. It would be feasible to extend this module to locate other parts of the body in addition to the face, and move onto full or partial human motion capture, or even gesture recognition. However, for the purpose of this work we are primarily interested in locating the face.

Face registration is the next stage of the process. This involves verifying that the target detected is indeed a face, and registering the locations of the facial features. We have not considered actual recognition in this thesis. However, if it

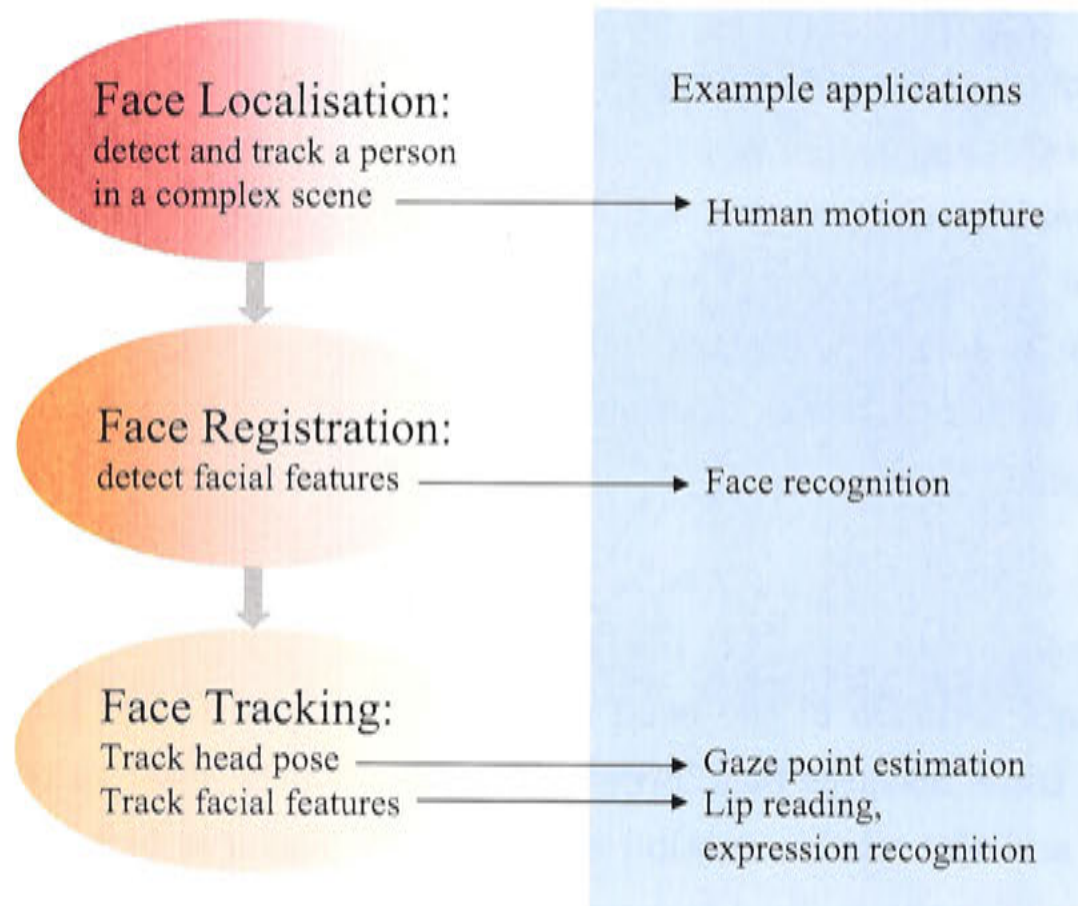


Figure 1.2: Overview of enabling a computer to “see” a face, and some typical applications associated with the different stages.

is desired to automatically recognise the face from a set of known faces, then the facial feature locations from the face registration module can be used to normalise the appearance of the face in preparation for the application of a face recognition algorithm.

Once facial features have been detected it is possible to track the pose of the head and track the relative locations of deformable facial features. This essentially captures all the information the face has to offer without determining a dense 3D model of the subject. From here it is feasible to perform lip tracking, automate a facial avatar, or attempt expression recognition.

This thesis will focus exclusively on capturing visual information describing the face, thus enabling a computer to “see” a face. Face localisation, registration, and tracking are each considered in turn and examples of implementations of each of these are presented. Fast and efficient visual cues are also considered in detail, and these cues are applied to the various detection and tracking tasks required.

Computer vision must be realtime to facilitate useful interaction with humans.

Consequently, the methods developed in this thesis have a strong emphasis on speed and efficiency. All algorithms were initially implemented in Matlab, however, some realtime implementations have been made using C++. Furthermore, while the Matlab implementations typically run quite slowly, the algorithms are efficient enough to run in realtime in C/C++.

1.2 Key Contributions

- Fast detection of radial symmetry — a valuable cue for detecting eyes and other radially symmetric features in images.
- A system to adaptively allocate computational resources and fuse cues for robust person tracking.
- Face detection algorithm for initialisation of a head tracking system.
- A monocular and a stereo 3D lip tracking system, both operating in conjunction with 3D head trackers to allow the subject's head freedom of movement whilst tracking.

1.3 Outline of Thesis

Chapter 2 discusses the application of computer vision for locating and tracking people, in particular the face, and reviews previous research in this area. In Chapter 3 a novel image based transform is presented that allows efficient computation of radial symmetry in realtime; this transform is a powerful visual cue for face detection and is used in the systems described in the Chapters 4 and 5. Chapter 4 presents a vision system that adaptively allocates computational resources over multiple cues to robustly track targets in 3D. In Chapter 5 a system is described that performs automatic detection of facial features for the process of face and gaze tracking. Chapter 6 explores the problem of tracking the face and deformable facial features such as the lips. Finally, Chapter 7 closes the thesis with a summary of the key findings and suggestions for further research.

An outline of each chapter is presented below.

1.3.1 Related Work

Chapter 2 reviews related work in the field. We discuss the physical qualities governing facial appearance, and consider visual cues suitable for detecting faces in images. We then move on to look at previous research relevant to locating a face (or other specified target) in a cluttered and dynamically changing environment, placing particular emphasis on the need to fuse multiple visual cues in order to obtain a robust estimate. Next we review previous work on face registration, that is, verification that a face is present and determining the location of facial features. Then we look at face tracking, both tracking of the head pose and tracking deformable facial features such as the mouth. Finally the chapter closes with a summary of the key points.

1.3.2 A Fast Radial Symmetry Transform

Chapter 3 presents a new image transform that utilizes local radial symmetry to highlight points of interest within a scene. Its low computational complexity and fast run-times make this method well suited for realtime vision applications. The performance of the transform is demonstrated on a variety of images and compared with leading techniques from the literature. Both as a facial feature detector and as a generic region of interest detector the new transform is seen to offer equal or superior performance to contemporary techniques at a relatively low computational cost. A realtime implementation of the transform is also presented demonstrating the effectiveness of the transform for highlighting peoples eyes in realtime.

1.3.3 Face Localisation

Chapter 4 considers the problem of *face localisation* in a complex, dynamic environment. A vision system is presented that adaptively allocates computational resources over multiple cues to robustly track a target in 3D. The system uses a particle filter to maintain multiple hypotheses of the target location. Bayesian probability theory provides the framework for sensor fusion, and resource scheduling is used to intelligently allocate the limited computational resources available across the suite of cues. The system is shown to track a person in 3D space moving in a cluttered environment. An additional example is shown demonstrating

how the system can be extended to track multiple targets, using multiple particle filters, and inhibition of returns to prevent different filters from locking onto the same target.

1.3.4 Face Registration

Chapter 5 examines the problem of *face registration*, that is, automatically detecting facial features and confirming the presence of a face in an image. A face registration system is presented that is designed to perform automatic detection of facial features for the purpose of face and gaze tracking, and hence provide the capability of face tracking without the requirement of a user specification or calibration stage. Motion information is used to detect blinks, indicating possible eye locations and an associated face candidate. Facial features (eyes, mouth corners, nostrils and eyebrows) are located and the face candidate is verified by examining the topology of these features.

1.3.5 Face Tracking

Chapter 6 explores the problem of tracking the face and deformable facial features such as the lips. In order to effectively track deformable facial features relative to an unconstrained head it is also necessary to track the head pose. In this chapter two case studies are presented, the first is a monocular lip tracker, and the second is a stereo lip tracking system that tracks the mouth shape in 3D.

Tracking the lips has a broad scope of applications across the field of human-computer interaction, including animation, expression recognition, and audiovisual speech processing. As people talk, their heads naturally move about as they gesture and follow conversation cues. It is necessary for a lip tracking system to be robust with respect to this behaviour; to be able to detect, monitor and account for movement of a speaker's head.

The mouth is a 3D feature which deforms in all spatial dimensions. In order to fully describe the mouth shape it is necessary to track it in 3D. Providing such a description of the mouth shape is essential for accurate 3D character animation, and also provides significantly more information for audio-visual speech processing and other human-computer interaction applications.

1.3.6 Conclusion

Chapter 7 closes the thesis with a summary of the key findings and achievements, and suggestions for further research.

1.4 Chapter Summary

This chapter has introduced and motivated the research reported in this thesis. We have discussed the importance of visual information in both interpersonal interaction between people and human-computer interaction, and stressed the point that a computer that can see people is significantly closer to a computer that we can interact with like we do with other human beings. The reader was then introduced to the problem of enabling a computer to see a person, and given a breakdown of a number of key elements of this problem. Finally we presented an overview of the research in this thesis, showing how it contributes towards solving the problem of enabling a computer to really “see” a person.

Chapter 2

Related Work

In the first chapter we discussed how the problem of enabling a computer to “see” a face can be broken down into face localisation, face registration and face tracking. This chapter reviews previous research in each of these three areas. The anatomy of the human face is also discussed along with visual cues suitable for detecting faces in images.

The first section of this chapter opens with a discussion of the physical qualities governing the appearance of a face, and reviews visual cues suitable for detecting faces in images. The following section reviews previous research relevant to locating a face (or other specified target) in a cluttered and dynamically changing environment; particular emphasis is placed on the need to fuse multiple visual cues in order to obtain a robust estimate. The third section reviews previous work on face registration, that is, verification that a face is present and determining the location of facial features. In the fourth section a brief background of face tracking is presented, this involves both tracking of the head pose and tracking deformable facial features such as the mouth. The chapter closes with a summary of the key points.

2.1 Cues for Person Tracking

2.1.1 The Human Face

Our faces are central to our identities as human beings. We recognise others and ourselves primarily from facial appearance. Four of the five senses — sight,

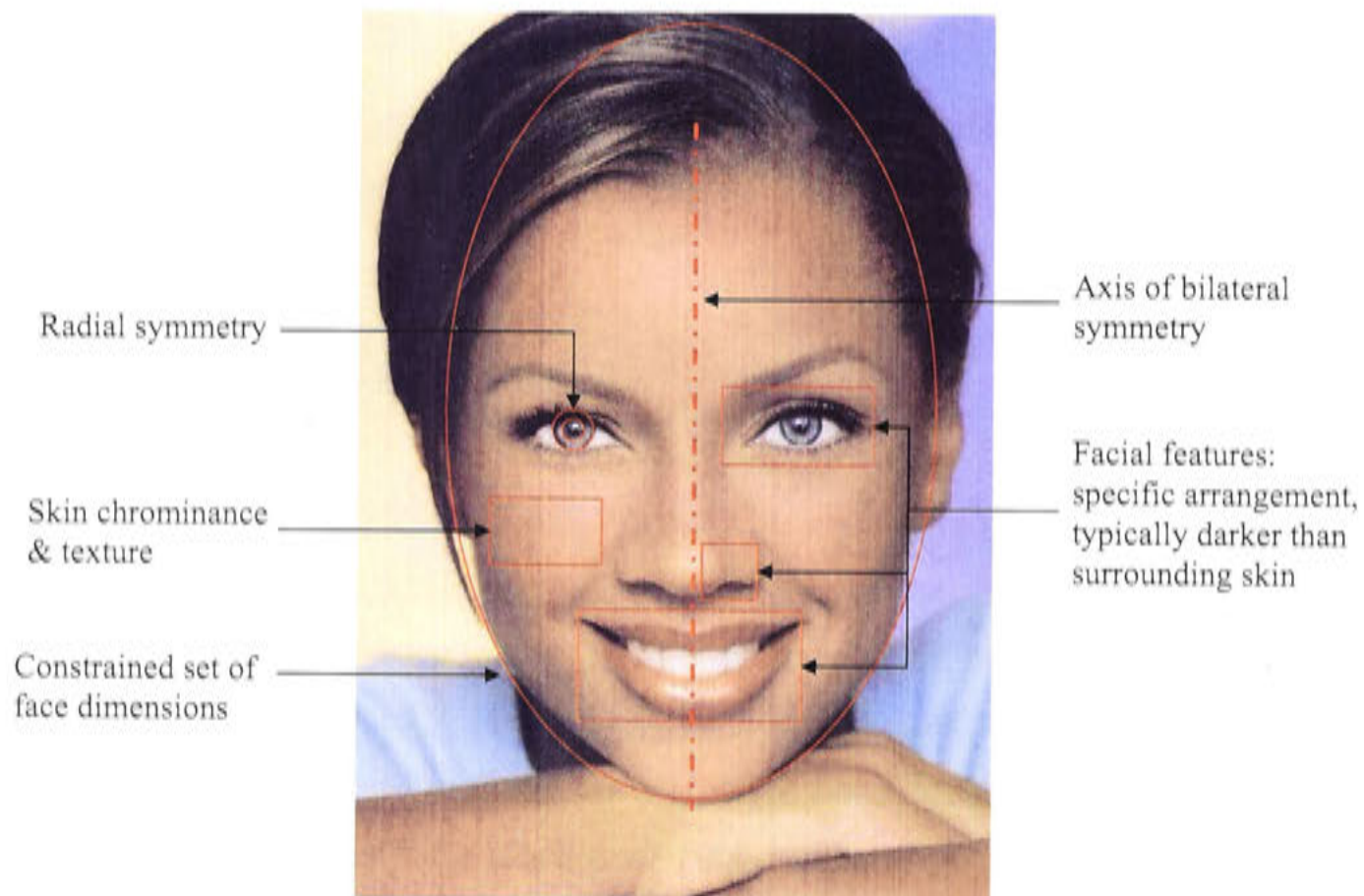


Figure 2.1: Facial qualities suitable for detection by a computer.

hearing, taste and smell — are perceived by organs within the facial region, and from another person's face we can sense how they are feeling, where their attention is focussed, and even make an educated guess as to whether they are lying or withholding information. With 7,000 discrete facial expressions (Bates and Cleese, 2001) at our disposal the face is rich with information, so it is not surprising that the face is our primary focus when we interact with others. Indeed, such interactions are often referred to as *face-to-face* encounters.

What qualities does the face have that makes it look like a face, and which of these qualities can be used by computer vision to allow us to automatically locate faces in images? Figure 2.1 shows a frontal view of a face with a number of visual attributes indicated that are suitable for detection by a computer vision system. The topology of the facial features is the face's most distinctive quality, that is, the arrangement of the eyes, nose and mouth, and the bilateral symmetry between the left and right sides of the face. The majority of the face is skin-coloured and of a smooth texture. Facial features such as the eyes, nostrils and mouth generally appear darker than the surrounding skin, and the irises and pupils of the eyes exhibit local radial symmetry. The size and dimensions of a face are also quite constrained, Pheasant (1986) presents a table of face dimensions of the general population of British adults aged 19-65 years, *circa* 1986, which is repeated in

Table 2.1.

Table 2.1: Face Dimensions of British Adults

Dimension	Men		Women	
	Mean (mm)	SD (mm)	Mean (mm)	SD (mm)
Head length	195	8	180	7
Head breadth	155	6	145	6
Maximum diameter of chin	255	8	235	7
Chin to top of head	225	11	220	11
Ear to top of head	125	6	125	8
Ear to back of head	100	7	100	9
Bitragion breadth	135	6	130	5
Eye to top of head	115	7	115	9
Eye to back of head	170	8	160	10
Interpupillary breadth	60	4	60	4
Nose to top of head	150	10	145	12
Nose to back of head	220	9	205	10
Mouth to top of head	180	9	170	11
Lip length	50	5	45	4

We desire our system to be able to detect anyone, regardless of race, sex, age or stature. With this in mind we look at head sizes from different populations, in an attempt to determine a range of head sizes within which every person will lie. Examining anthropometric data from Pheasant (1986) for males and females from North American, British, French, Swiss, German, Swedish, Polish, Japanese, Hong Kong Chinese and Indian populations we find the American male has the largest adult head size, and the smallest is that of Indian women. Thus we have a range within which we expect adult head sizes to fall. Including children in the search space will lead to a broader range of acceptable head sizes, however, if we restrict ourselves to only searching for children above five years old this only slightly extends the acceptable range of head sizes. Table 2.2 presents the head dimension of these bounding populations. Note that only British data was considered for children.

Table 2.2: Head Dimensions Bounding Populations

Population	Head length		Head breadth	
	Mean	SD	Mean	SD
Newborn infants (British)	120	4	95	3
5 year old girls (British)	165	8	130	5
Smallest adult (Indian female)	170	7	135	5
Largest adult (American male)	195	8	155	6

It is possible to calculate an *average face* by overlaying numerous face images with

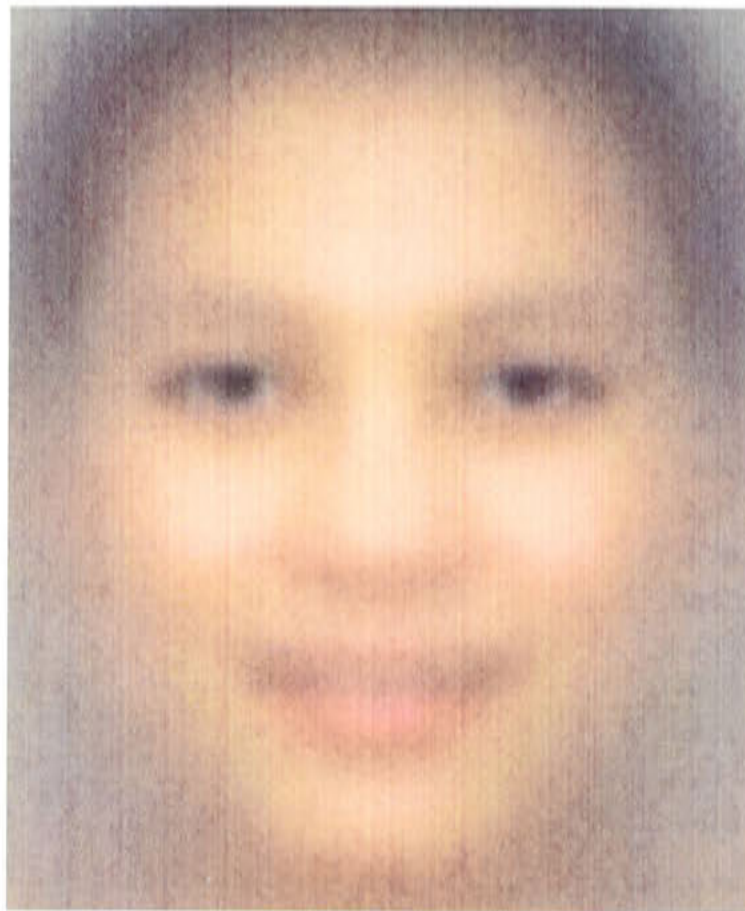


Figure 2.2: Average face.

the facial features aligned. Average faces have been used previously in computer vision to search for faces in images (Cai and Goshtasby, 1999), and in studies of human facial beauty (Grammer and Thornhill, 1994). However, these average faces have typically been constructed from a modest number of faces (Cai and Goshtasby used 16, and Grammer and Thornhill (1994) constructed male and female average faces using 44 and 52 subjects respectively). We have constructed an average face from 224 images of faces of men and women of different races obtained from the internet. Each image was rotated so the eyes were horizontal, and warped so the interpupillary distance and the distance from the mouth to the eyes were the same across all images. The ratio of the interpupillary distance to mouth to eye distance was determined by averaging the male and female populations in Table 2.1 (giving a ratio of 1:1). The resulting average face is shown in Figure 2.2. The average face provides a useful reference for designing cues to detect faces and facial features in images.

There are a number of different problems to consider when looking for a face. Knowing the range of acceptable head sizes allows us to search for head-sized blobs using stereo depth information, and identify regions of motion that could potentially be heads. Face-sized regions of skin colour can also be identified, as can peaks in radial symmetry and dark blobs that could be facial features. Searching for regions with bilateral symmetry and features clustered in face-

like arrangements is difficult to do efficiently and robustly, however, these facial qualities are useful to check when it comes to verifying whether or not a detected target is a face.

The remainder of this section reviews previous research in this area, covering skin detection, depth map estimation, motion detection, and radial symmetry detection.

2.1.2 Skin Detection

Detecting skin regions is a first step in the majority of recent face detection methods. The key quality that differentiates skin from non-skin regions in images is *colour*. Colour has been successfully used to identify regions of human skin in images in numerous applications. Interestingly, human skin colour varies little between different races. The primary variation is in its intensity, that is proportional to the amount of melanin in the skin.

Swain and Ballard (1991) demonstrated that the intersection of colour histograms in colour space could be used to reliably identify coloured objects. However, this technique was sensitive to colour intensity and thus the ambient light source. Several years later Hunke (1994), Hunke and Waibel (1994) and Schiele and Waibel (1995) developed a skin colour detector which was invariant with respect to intensity. They modelled colour in a two dimensional chrominance space¹ obtained by normalising the RGB colour space with respect to intensity (see equations 2.1 and 2.2). Since this time a plethora of different skin colour detection schemes have been reported in the literature.

The general approach of skin colour segmentation schemes is summarised as follows.

Initially a skin colour model is built off-line, this involves:

- Sample colour images containing only skin colour are passed to the system (these are typically in RGB format), Figure 2.3(a).
- The colour value of every pixel is mapped to a two dimensional chrominance space (some schemes map to a colour space with three dimensions, but these

¹A *chrominance space* is a two dimensional space generated by removing the intensity component from a three dimensional colour space such as RGB or HSV.

are in the minority) to form a skin colour histogram. Figure 2.3(b).

- A model is selected describing the distribution of skin colour pixels in chrominance space. Figure 2.3(c).

Testing images for skin colour is done on a pixel-by-pixel basis as shown in Figure 2.4:

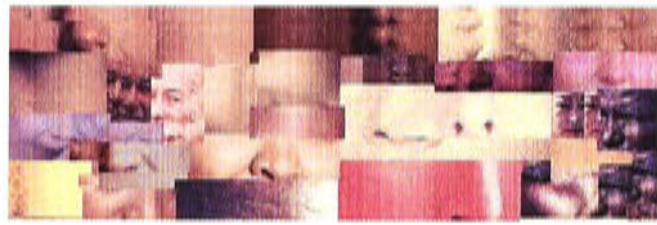
- The colour information is converted to the appropriate chrominance space.
- The skin-likeness of each pixel is determined by the value of the skin colour distribution function corresponding to the pixel's location in chrominance space.

A threshold is generally applied to the output to produce a binary image of skin-coloured regions, however, pixels can be left as grey-levels giving a continuous measure of how “skin-like” they appear.

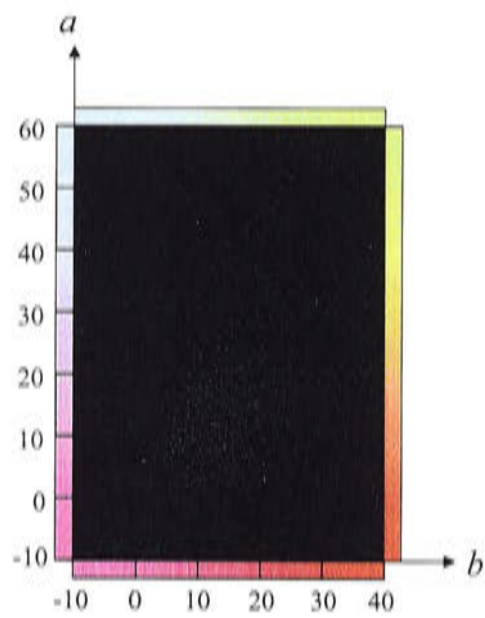
The main differences between different skin colour detection schemes are the chrominance space chosen, and the distribution used to model the skin in chrominance space.

The effectiveness of a skin detection algorithm depends on the appropriateness of the chrominance space in which the skin chroma is modelled. It is desirable to use a space in which the skin chroma distribution can be accurately modelled and segmented from non-skin chroma. Just about every colour space (or corresponding chrominance space) has been used for skin colour detection, examples include RGB (Sato *et al.*, 1997), normalised rg (Hunke, 1994; Kumar and Poggio, 2000), HSV (Sobottka and Pitas, 1996a), CIE (Commission Internationale de L'Éclairage) XYZ (Wu *et al.*, 1999), CIE LUV (Yang and Ahuja, 1998), and CIE Lab (Cai and Goshtasby, 1999).

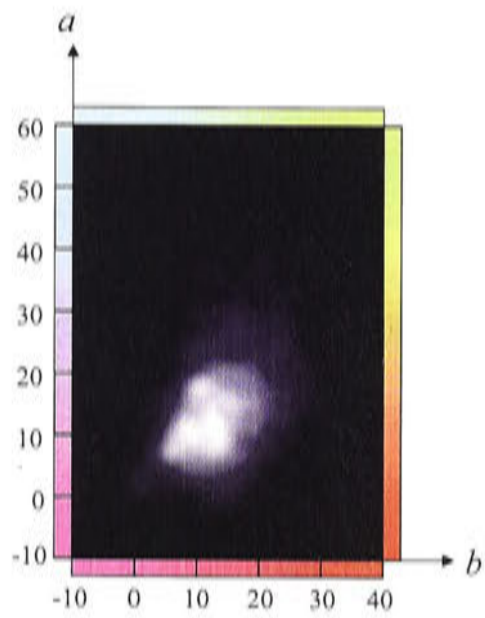
Two recent studies have compared the performance of different colour spaces for human skin detection (Terrillon and Akamatsu, 1999; Zarit *et al.*, 1999), whilst these studies fail to agree on an optimal colour space, results from both studies support the HS chrominance space as exhibiting the smallest overlap between skin and non-skin distribution. Terrillon and Akamatsu (1999) examine the comparative performance of nine different colour spaces applied to detecting Asian and Caucasian faces in complex images using a single multivariate Gaussian



(a)



(b)



(c)

Figure 2.3: Constructing a skin colour model. (a) Image of multiple skin samples. (b) Plot of chrominance values in ab chrominance space, from Cai and Goshtasby (1999). (c) Example skin chrominance model in ab space, from Cai and Goshtasby (1999).

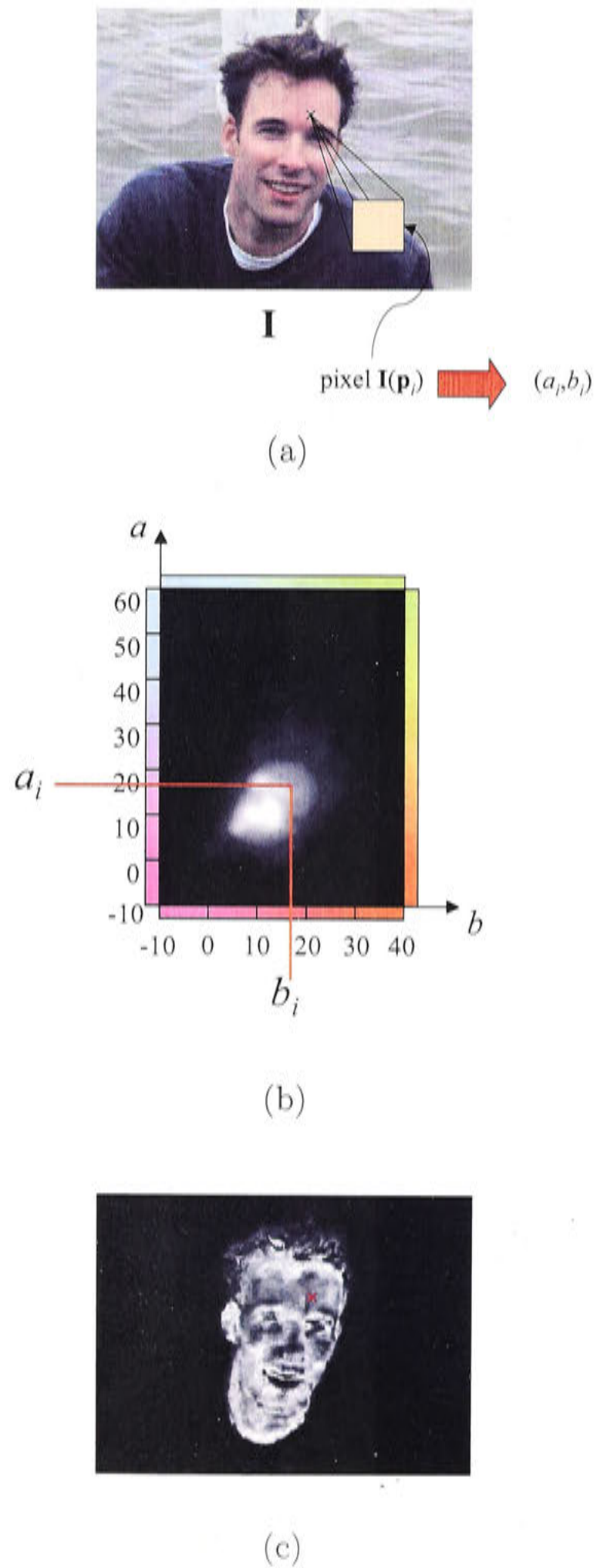


Figure 2.4: Detecting skin. (a) Input image, convert to appropriate chrominance space. (b) Determine skin-likeness of each pixel from skin model. (c) Result showing skin-likeness of each pixel in input image.

skin colour distribution model. They test normalised rg , CIE-xy, TS, CIE-DSH, HSV, YIQ, YES, CIE LUV and CIE LAB and conclude that their own TS chroma space (Terrillon *et al.*, 1998) designed especially for this purpose shows the best results, followed by the normalised rg space. Zarit *et al.* (1999) compare the performance of CIE LAB, Fleck HS, HSV, normalised rg and YC_rC_b with two different skin colour modeling schemes and conclude that HSV and Fleck HS provide superior performance.

From these studies on classification performance, normalised rg , HS and TS chrominance spaces appear are the most effective for skin segmentation. However, classification performance is not the only factor that needs to be taken into consideration. The computational load of converting to different chrominance spaces is also an important factor when choosing a chrominance space for realtime skin detection.

Video cameras generally deliver raw colour image information to a computer in YUV format, where Y is a full resolution luminance channel and U and V are chrominance channels, with one value for every two pixels. These are converted to standard RGB format for storing in memory and displaying on the screen, and as a result the majority of colour conversions consider RGB as the base colour type. The normalised rg chroma, for instance, are calculated from the RGB colour values using,

$$r = \frac{R}{R + G + B} \quad (2.1)$$

$$g = \frac{G}{R + G + B} \quad (2.2)$$

The TSL space which leads to the TS chroma is defined as (Terrillon and Akamatsu, 1999)

$$S = 3\sqrt{\frac{r'^2 + g'^2}{5}}$$

$$T = \begin{cases} \frac{1}{2\pi} \tan^{-1}(r'/g') + \frac{1}{4} & \text{if } g' > 0 \\ \frac{1}{2\pi} \tan^{-1}(r'/g') + \frac{3}{4} & \text{if } g' < 0 \\ 0 & \text{if } g' = 0 \end{cases}$$

$$L = 0.299R + 0.587G + 0.114B$$

where $r' = r - \frac{1}{3}$ and $g' = g - \frac{1}{3}$, and r and g are defined by Equations 2.1 and 2.2.

However, there is no reason why the raw UV chrominance information cannot be used for skin segmentation. The YUV colour format provides us with a pre-calculated chrominance image that requires no additional computation to generate.

The second key element in a skin-colour extractor is the model used to represent the skin colour distribution in chrominance space. Such models range from primitive rectangular regions achieved by thresholding of chrominance values (Sobotka and Pitas, 1996b) to empirical histogram look-up tables (Hunke and Waibel, 1994) and sophisticated probabilistic and statistical models (Yang *et al.*, 2000; Wu *et al.*, 1999).

Some researchers (Yang and Waibel, 1996; Yang and Ahuja, 1998) have hypothesised that all skin colour – regardless of race – can be satisfactorily modelled by a single multivariate Gaussian distribution. It is true that skin values in chrominance space deviate little due to race, however, some subjects do exhibit slightly different skin chrominance distributions independently of race (Omara, 2000). This observation has led to a number of more complex skin models, examples include multiple Gaussian distributions (Omara, 2000), fuzzy modelling techniques (Wu *et al.*, 1999) and neural network based designs (Chen and Chiang, 1997).

Cai and Goshtasby (1999) proposed a simple numerical technique for building a skin colour look-up table in chrominance space. The result is effectively a numerical approximation of a complex multi-Gaussian model, and is obtained by convolving the chroma histogram with a Gaussian to make a “skin cloud” in chroma space. This approach is very attractive as it offers a diverse and accurate model with the speed of an empirical lookup table. The drawback is that it is difficult to adapt the skin model to changing lighting conditions, since it is not represented as a formal statistical distribution function. This sensitivity to lighting conditions is the main shortcoming of skin colour detection schemes, and while the use of intensity invariant chroma spaces has reduced this sensitivity, it is still a problem.

Some researchers have considered adapting skin colour models to varying lighting conditions. Yang and Waibel (1996) and Yang *et al.* (1998b) showed how to

modify the parameters of their Gaussian model to adapt to changes in lighting during operation. Raha *et al.* (1998) used Gaussian mixture models to detect skin colour, hair, and clothing and presented a technique for dynamically updating these models to account for changing lighting conditions. Sigal and Sclaroff (2000) use a Hidden Markov Model to evolve a skin colour distribution model in HSV colour space, and claim their system reliably extracts skin under widely varying lighting conditions – including multiple sources of coloured light.

An alternative approach is to simply build the original chrominance histogram using samples from all lighting conditions under which the system is intended to operate. These conditions cannot be too diverse or the histogram could potentially contain *all* possible colours, however, for a constrained set of lighting conditions this is a feasible approach.

We base our approach to skin detection on that of Cai and Goshtasby (1999) which offers a fast, efficient and simple method that delivers a high level of performance. However, we augment the method by building a three-dimensional skin colour histogram to better discriminate across varying lighting conditions. Also, rather than using CIE Lab colour space we use YUV since these channels are available directly from our cameras and saves performing the additional non-linear conversion to CIE Lab space.

2.1.3 Depth Maps

Stereo images have long been used for calculating depth in computer vision applications (Jarvis, 1983). There are other means of estimating depth that do not require stereo, such as using a single camera and varying the focus, estimating structure from motion, or even shape from shading. However, stereo is by far the most popular and robust method of estimating depth in the near field; indeed stereo is a strong cue for human depth perception for distances up to 10 meters. Figure 2.5 shows an example of a pair of stereo images and a depth map generated from these images. The generation of such depth maps is discussed below.

Stereo imaging is best illustrated using the pinhole camera model to represent the cameras involved. This model is shown in Figure 2.6 (a) and demonstrates how each pixel in the image corresponds to a ray in 3D space — so an object visible at a particular image point could lie anywhere on the ray through that point (beyond the image plane). Now consider the case shown in Figure 2.6 (b),

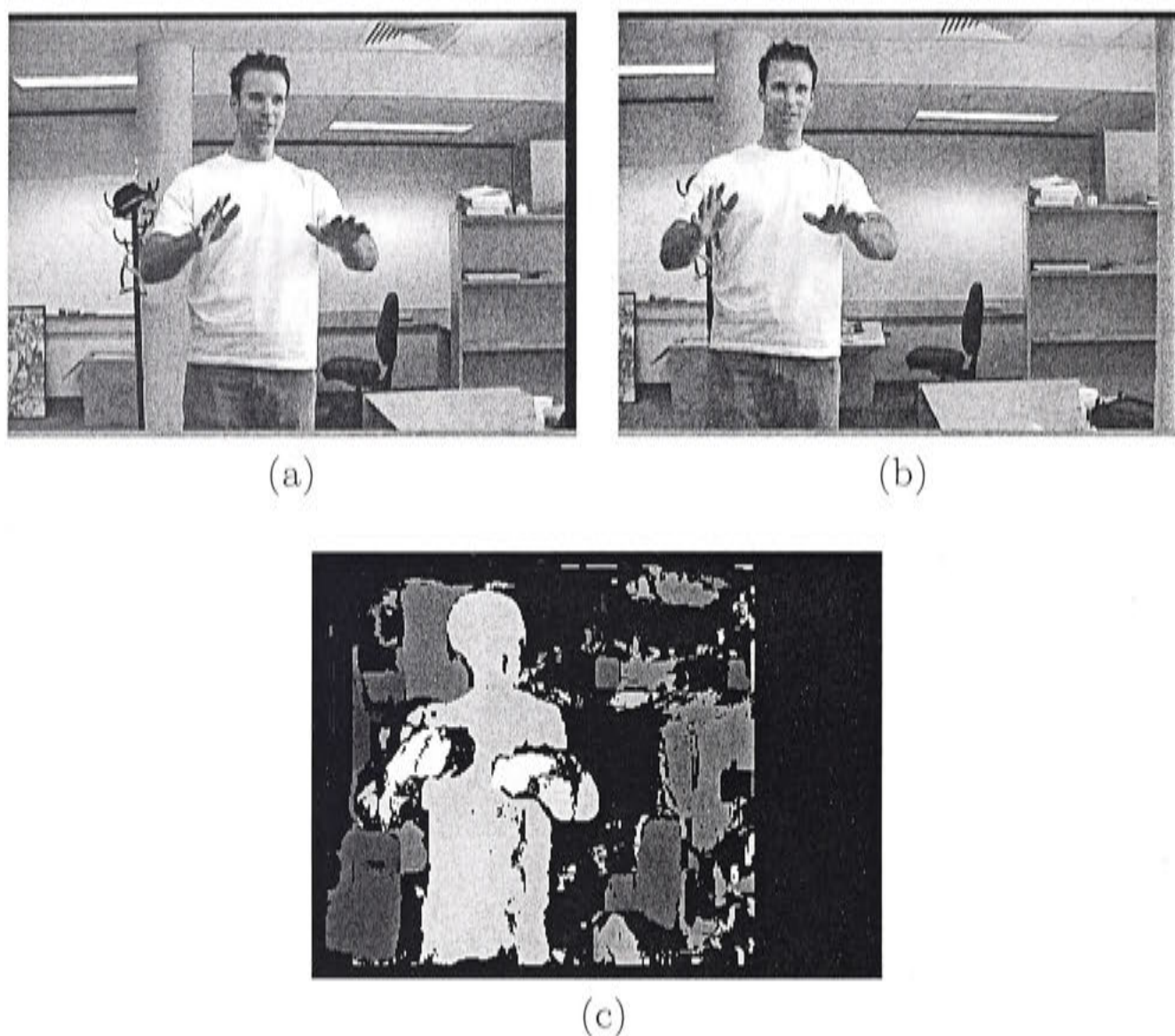


Figure 2.5: A stereo image pair and associated depth map, courtesy of Luke Fletcher. (a) Left image. (b) Right image. (c) Depth map with lighter values indicating shallower depths.

where two cameras are looking at same point. An object observed by camera A lies on a ray that appears as a line in camera B. This is called an epipolar line, and is dependent entirely on the epipolar geometry of the cameras, that is, the location and orientation of the cameras with respect to each other, and the internal parameters of the cameras. The epipolar geometry is independent of the objects in front of the camera, so regardless what images are observed, a particular image location will always correspond to the same epipolar line in the other camera view. All epipolar lines radiate out from a fixed image point called the epipole, which is the image of the centre of the other camera, as shown in the figure.

When we are computing depth maps we are essentially just computing a series of point correspondences between the two images. So given a point in image A we need only attempt to locate this point along the corresponding epipolar line in

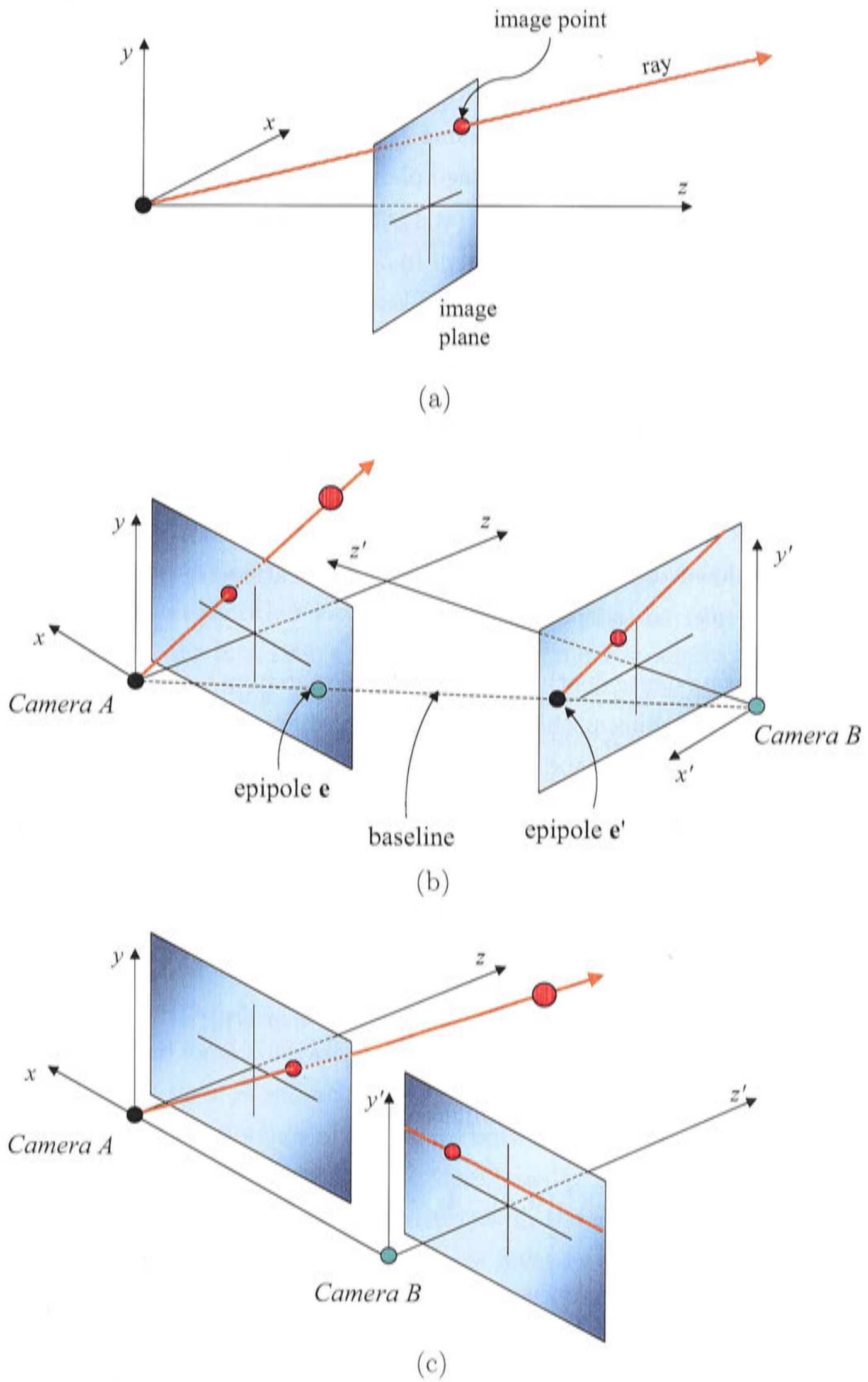


Figure 2.6: Pinhole camera model and stereo camera configurations. (a) Single camera. (b) Verging stereo cameras. (c) Aligned stereo cameras.

image B. A straightforward expression for determining the epipolar lines can be determined by calculating the epipolar geometry and determining the *fundamental matrix* (Hartley and Zisserman, 2000). However, this is not necessary if we set up the cameras in an aligned configuration as shown in Figure 2.6 (c). This requires both cameras to share the same X-axis (or alternatively Y-axis), have parallel optical axes, and coplanar image planes. calculating a stereo depth map. In this configuration an image point at height y in one image will correspond to a horizontal epipolar line at height y in the other image. Since the cameras are directly side-by-side the epipoles are located at infinity, hence the parallel epipolar lines.

The depth of an object observed in two stereo images from calibrated aligned cameras can be determined from the *disparity*² between the object's location in the two images. The problem of determining the 3D depth map, such as the one shown in Figure 2.5(c), (or equivalently the disparity values) from a pair of stereo images comes down to finding the corresponding locations of points in the both images, this is referred to as *stereo matching*.

When constructing dense depth maps area-based matching techniques are used to solve the stereo matching problem. A number of different area-based techniques are available (Aschwanden and Guggenbuhl, 1993). Denoting the template window as \mathbf{I}_1 , the candidate window as \mathbf{I}_2 , the mean pixel values of these windows as $\bar{\mathbf{I}}_1$ and $\bar{\mathbf{I}}_2$ respectively, and summation over the window as $\sum_{(u,v) \in W}$, these are:

- Sum of Absolute Differences,

$$\sum_{(u,v) \in W} |\mathbf{I}_1(u, v) - \mathbf{I}_2(x + u, y + v)|$$

- Zero mean Sum of Absolute Differences,

$$\sum_{(u,v) \in W} |(\mathbf{I}_1(u, v) - \bar{\mathbf{I}}_1) - (\mathbf{I}_2(x + u, y + v) - \bar{\mathbf{I}}_2)|$$

- Sum of Squared Differences,

$$\sum_{(u,v) \in W} (\mathbf{I}_1(u, v) - \mathbf{I}_2(x + u, y + v))^2$$

² *Disparity* refers to the shift of a 3D object's position in an image when the camera is moved perpendicular to the optical axis.

- Zero mean Sum of Squared Differences,

$$\sum_{(u,v) \in W} ((\mathbf{I}_1(u, v) - \bar{\mathbf{I}}_1) - (\mathbf{I}_2(x + u, y + v) - \bar{\mathbf{I}}_2))^2$$

- Normalised Cross Correlation,

$$\frac{\sum_{(u,v) \in W} \mathbf{I}_1(u, v) \cdot \mathbf{I}_2(x + u, y + v)}{\sqrt{\sum_{(u,v) \in W} \mathbf{I}_1(u, v)^2 \cdot \sum_{(u,v) \in W} \mathbf{I}_2(x + u, y + v)^2}}$$

- Zero mean Normalised Cross Correlation,

$$\frac{\sum_{(u,v) \in W} (\mathbf{I}_1(u, v) - \bar{\mathbf{I}}_1) \cdot (\mathbf{I}_2(x + u, y + v) - \bar{\mathbf{I}}_2)}{\sqrt{\sum_{(u,v) \in W} (\mathbf{I}_1(u, v) - \bar{\mathbf{I}}_1)^2 \cdot \sum_{(u,v) \in W} (\mathbf{I}_2(x + u, y + v) - \bar{\mathbf{I}}_2)^2}}$$

Regardless of which method is used, generating a dense depth map across an entire image is a computationally expensive procedure, as each image location must be matched with every other location on the corresponding epipolar line in the second image. In the late 1990's Konolige (1997) and Kanade *et al.* (1996) both demonstrated systems able to generate dense depth maps in realtime, however, these systems relied on specialised hardware. In 2000 Kagami *et al.* presented a method for efficiently generating dense depth maps in realtime without requiring specialised hardware. This was achieved by using four key techniques: recursive normalised cross correlation, cache optimisation, online consistency checking, and use of the Intel MMX/SSE(R) instruction set.

Preprocessing of images before performing stereo matching can increase the effectiveness of the matching process. Preprocessing typically involves filtering images to increase local contrast, and is particularly advantageous for matching areas with low texture. Standard linear filtering approaches used are Laplacian of Gaussian (LoG), or Difference of Gaussian, both of which increase local contrast in the image. The LoG is the sum of the Gaussian's second derivatives. Figure 2.7 shows a Gaussian, the first derivatives in the x and y directions and the LoG, $\nabla^2 \mathbf{G}$. Applying a LoG across an image involves convolving the LoG kernel $\nabla^2 \mathbf{G}$ across the image. Unfortunately this kernel is non-separable, therefore the convolution cannot be split into two one-dimensional convolutions, and is of order $O(KN^2)$ for an $N \times N$ kernel across an image with K pixels.

However, a LoG kernel can be closely approximated by the more efficient Difference of Gaussian (DoG) filter. As its name implies, a DoG kernel is constructed as

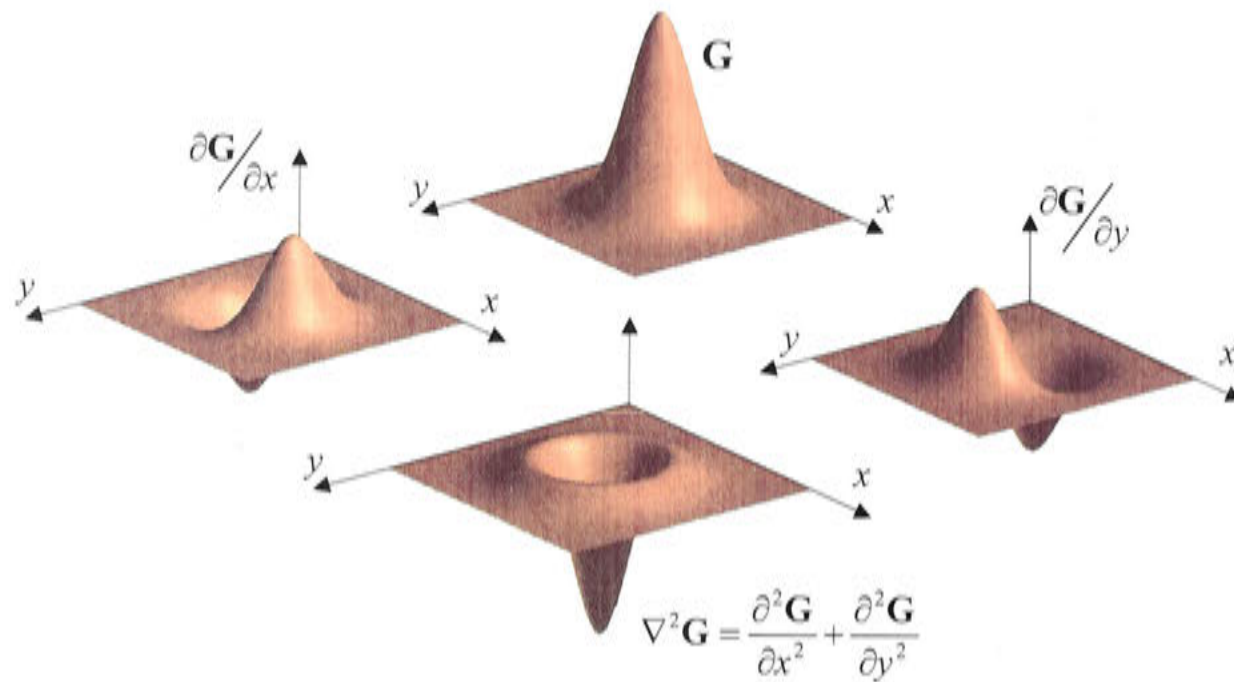


Figure 2.7: Laplacian of Gaussian. From top to bottom: Two-dimensional Gaussian kernel, derivatives of Gaussian in x and y directions, and Laplacian of Gaussian.

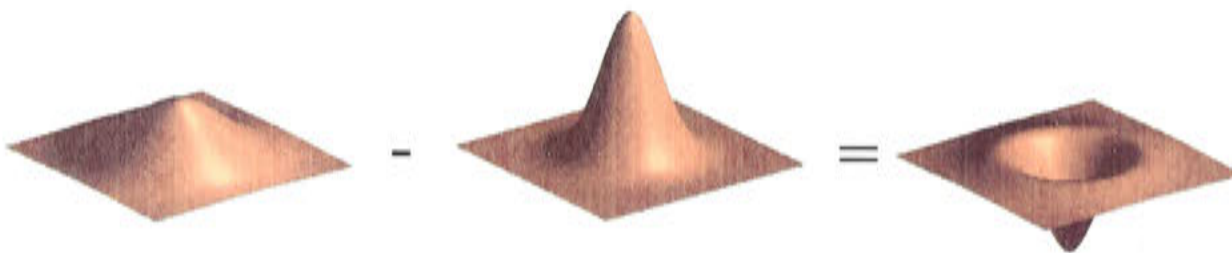


Figure 2.8: Difference of Gaussian kernel is generated as the difference of two Gaussians.

the difference of two Gaussian kernels as shown in Figure 2.8. Applying the DoG filter is more efficient than the LoG since each Gaussian can be applied separately as two one-dimensional convolutions, and the results subtracted to determine the DoG response.

Zabih and Woodfill (1994) present two non-parametric local transforms especially formulated for enhancing the computation of visual correspondences, these are called the *rank* and *census* transforms. The effectiveness of these transforms for generating dense depth maps in realtime was demonstrated by Banks *et al.* (1997) who applied the rank and census transforms when generating depth maps for an underground mining application.

The rank transform is calculated for a pixel p by counting the number of pixels in a local region centred on p whose intensities are darker than the intensity at p . For example, Figure 2.9 shows a 3×3 local region centred at a point p , with

1	1	0.3
1	p 0.6	0
0.6	0.3	0

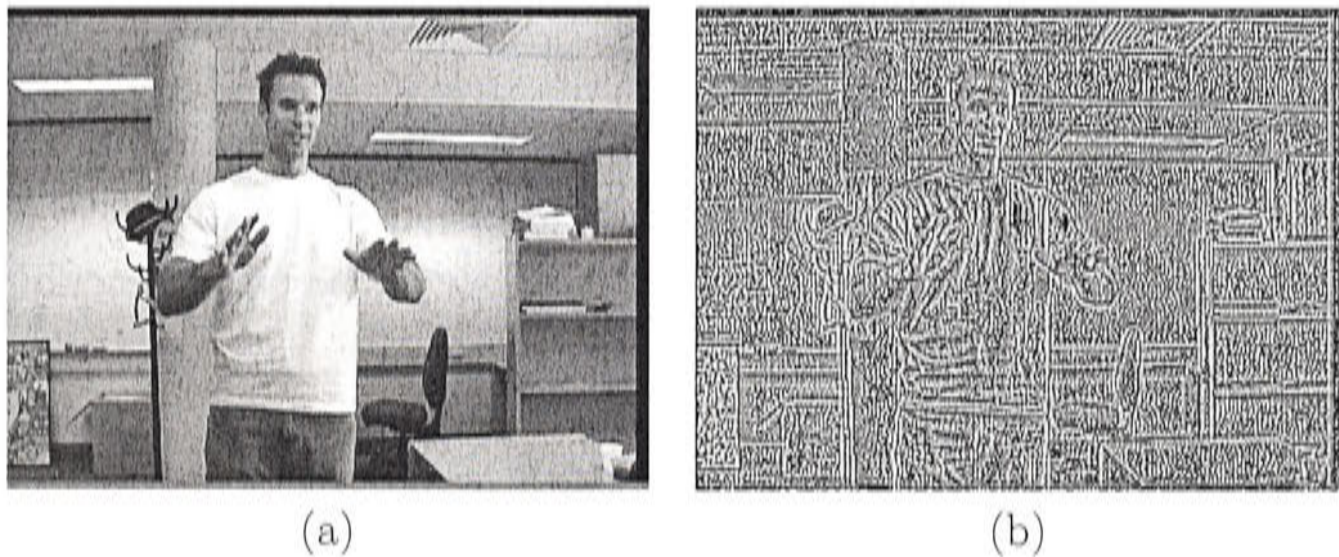
Figure 2.9: Example of a 3×3 neighbourhood centred on a point p .

Figure 2.10: Result of Zabih and Woodfill's rank transform with radius 1. (a) Original image. (b) Rank transform.

the intensities of the pixels indicated. The value of the rank transform at point p is 4, since there are 4 pixels darker than p in the local region. Applying this transform to an image, as shown in Figure 2.10, results in an increase in local texture, and since this texture will be consistent across both images of a stereo pair it can be used for stereo matching. It is particularly beneficial for matching in featureless areas of the image.

The census transform is an extension to the rank transform. Again the value at pixel p is determined by examining the pixels in a local region centred on p and determining which ones have intensities that are darker than the intensity at p . However, rather than simply counting how many of these there are, the census transform uses a binary code to record the locations of the pixels that were darker than p . Each location in the neighbourhood of p is assigned a position in a binary string, and if the pixel at this location is darker than p then the associated element in the binary string is set to 1, otherwise it is set to 0. For instance, determining the census transform over a 3×3 neighbourhood would require an 8-bit binary number to indicate which of the 8-elements surrounding the centre pixel were darker than the centre value and which were not.

While the census transform can provide useful structural information that can enhance stereo matching it is questionable that these enhancements are sufficient to warrant the significant additional computation required to compute the transform. On the other hand, preprocessing images with the rank transform or a Difference of Gaussian filter prior to matching is relatively cheap computationally and the quality of the depth maps generated benefit from the improved matching results. Of these two operators the Difference of Gaussian can be more efficiently implemented in software, whereas the rank transform is best suited to hardware implementation.

We use depth maps generated in realtime by the method of Kagami *et al.*. For maximum efficiency pre-filtering is done in software with a Difference of Gaussian filter, and stereo matching will be done using Sum of Absolute Differences.

2.1.4 Motion

There are several different methods for identifying regions of motion and segmenting moving objects in image sequences: image differencing, adaptive background subtraction, and optical flow. Figure 2.11 shows an example of each of these.

The simplest approach is *image differencing* (Figure 2.11(c)). Here corresponding pixel locations in two images are compared and locations where a significant change is observed are marked as regions of motion. This approach provides an efficient and straightforward means of locating potential regions of motion, however, since it is simply identifying pixels whose values have changed between the two images it is easily fooled by changes in lighting, camera position, or camera parameters (such as zoom). It also tends to detect shadows as areas of motion. Despite its shortcomings, the efficiency and effectiveness of this approach have found it used in many applications, particularly surveillance systems where the background is often stationary. Crowley and Berard (1997) used image differencing for estimating head location and localising blink positions in order to determine eye locations, and Bala *et al.* (1997) also detected blinks in this way. Image differencing is well suited for blink detection. Humans typically blink very rapidly — Hakkanen *et al.* (1999) reported a mean blink duration of 51.9 ms — so the transition from open to closed eyes can occur in the time between consecutive frames, making it impractical to explicitly track the closing and opening movement of the eyelids.

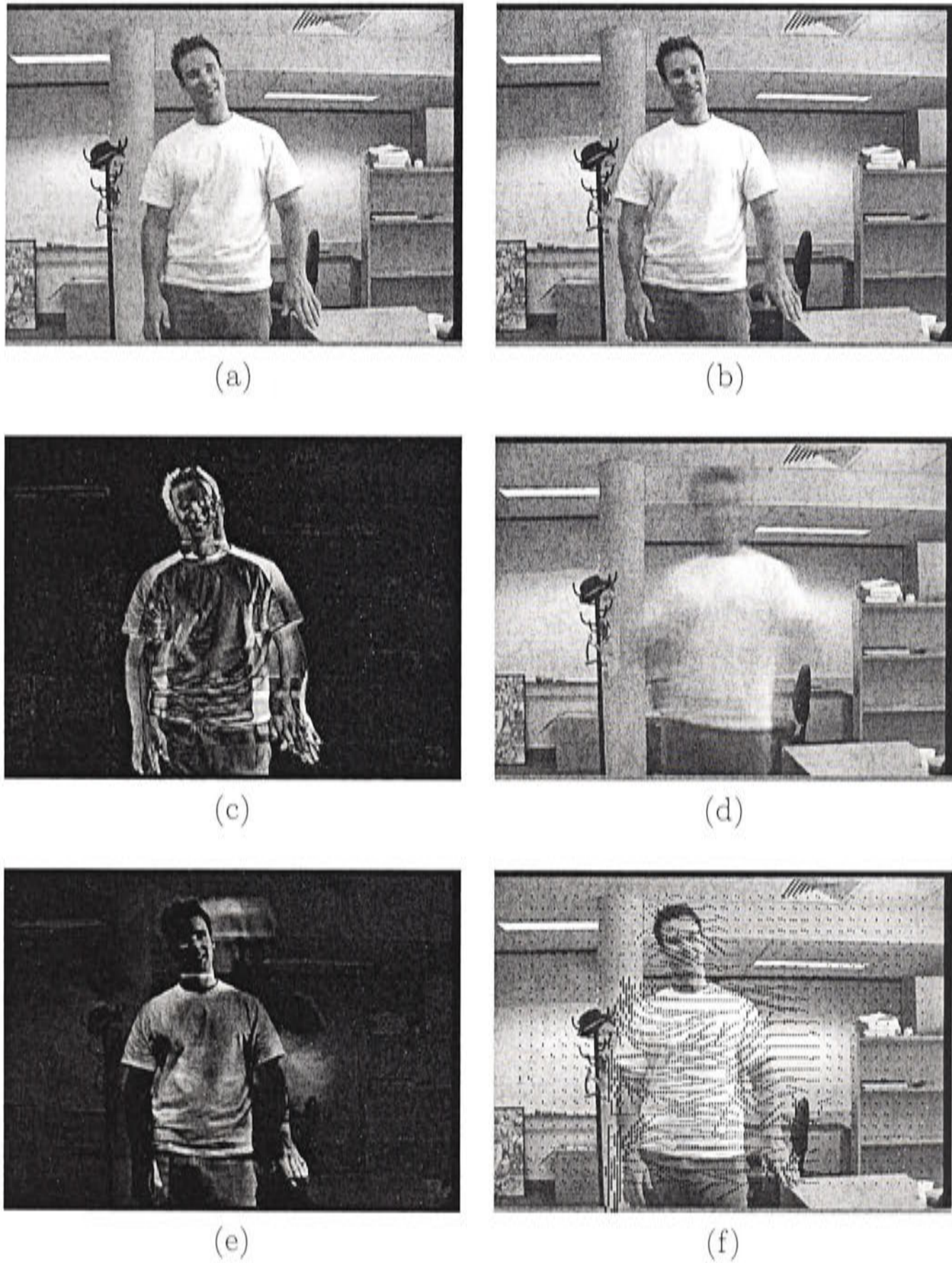


Figure 2.11: Two consecutive images in a 30Hz motion sequence and examples of different motion cues calculated from these and previous frames. (a) Previous frame. (b) Current frame. (c) Difference image. (d) Adaptive background. (e) Difference from adaptive background. (f) Optical flow, courtesy of Luke Fletcher.

Background subtraction is an extension of image differencing. Rather than differencing frames separated by a certain time delay, an image of the background is subtracted from the current image to highlight objects that were not present in the original background image. This method is very effective if a suitable background image is available, however, unfortunately this is often not the case. Even if it is feasible to capture an image of the background without the subject present, background subtraction will only be effective if the background remains static, and the lighting, camera and camera parameters all remain the same. For most applications it is unreasonable to expect the background to remain static throughout an image sequence, and so to overcome this problem adaptive background models have been developed. These allow a model of the background to be constructed and updated to accommodate changes in lighting and variations in the background.

Adaptive background subtraction provides a better measurement of motion than simple background subtraction. Whereas the latter simply differentiates between objects and the background, adaptive background subtraction highlights pixels that have changed recently in the image sequence (see for example Figure 2.11(e)). The adaptive background image (Figure 2.11(d)) is initialised as the current frame and updated each frame to be a weighted sum of itself and the current frame. Let \mathbf{A}_t be the adaptive background image at time t , and \mathbf{I}_t be the input image, then a motion image \mathbf{M}_t is defined as

$$\mathbf{M}_t = |\mathbf{I}_t - \mathbf{A}_t| \quad (2.3)$$

and each frame \mathbf{A}_t is updated as

$$\mathbf{A}_t = k\mathbf{I}_t + (1 - k)\mathbf{A}_{t-1} \quad (2.4)$$

where $k \in (0, 1)$. Like regular background subtraction this method is best suited to fixed cameras where the majority of the image remains constant, so motion of objects in the scene can be easily detected.

Collins *et al.* (2000) used this adaptive background approach in conjunction with image differencing to segment moving objects from a predominantly stationary background in an outdoor surveillance scenario. The adaptive background method is fast and efficient to compute, and the time it takes for stationary objects to be absorbed into the background can be modulated simply by varying the constant k in Equation 2.4.

A more sophisticated method for quantifying motion in images is *optical flow*, illustrated in Figure 2.11(f), which aims to directly measure the movement of pixels in an image sequence. An optical flow field is a vector field describing the direction of local motion at each point in the image. There are several approaches available for calculating optical flow, and Baron *et al.* (1994) provide a detailed review of different methods. Broadly, the techniques can be divided into correlation and constraint-based methods.

Correlation-based methods identify local motion by locating groups of pixels from the previous image in the current image. This involves searching over small 2D regions centred about where the pixels occurred in the previous image. It is computationally intensive, but conceptually simple, and can be implemented recursively to increase the efficiency. In 1999 Kagami *et al.* demonstrated realtime flow generation using a recursive method to calculate correlations, along with cache optimisation, and the Intel MMX instruction set (Kagami *et al.*, 1999).

Constraint-based optical flow methods (Horn and Schunk, 1981; Lucas and Kanade, 1981) rely on the *optical flow constraint equation*,

$$-\frac{\delta I}{\delta t} = u \frac{\delta I}{\delta x} + v \frac{\delta I}{\delta y} \quad (2.5)$$

where $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$, the derivation of which is included in Appendix B. Each element of this equation can be determined directly from the image sequence, $\frac{\delta I}{\delta x}$ and $\frac{\delta I}{\delta y}$ are the regular image derivatives describing how intensity changes across the image in the x and y directions, and $\frac{\delta I}{\delta t}$ indicates how fast the intensity is changing with time. By itself this one constraint equation is insufficient to solve for the two unknowns u and v . Horn and Schunk (1981) applied an additional global smoothness constraint, and Nagel (1983) offered a variation on this designed to better handle occlusion by not imposing the smoothness across strong intensity gradients. Lucas and Kanade (1981) presented an alternative approach that determined a weighted least squares solution to Equation 2.5 over a small spatial neighbourhood in the image.

Compared to image differencing and adaptive background subtraction, optical flow can potentially provide more useful information for identifying objects in dynamic scenes. For instance, it allows moving objects to be segmented from moving backgrounds. The main drawbacks are the coarseness and inaccuracies of the flow field that is typically produced, and the high computational requirement.

However, recent fast flow generation results (Kagami *et al.*, 1999) mean that optical flow is now a viable option for realtime tracking systems.

We use image differencing to detect blinks and locate eye positions as it is unquestionably the simplest and fastest method available. Adaptive background subtraction shall be used to help detect targets moving in cluttered scenes. Whilst this method essentially relies on the target moving and the rest of the scene remaining more-or-less static, it is extremely effective in this situation, and when combined in a multi-cue system will be able to provide complementary information when the target is undergoing motion. Optical flow is computationally expensive, and despite the recent work of Kagami *et al.* (2000) typically provides too sparse a flow field to warrant its inclusion in a multi-cue face localisation system.

2.1.5 Radial Symmetry Operators

A number of context-free attentional operators have been proposed for automatically detecting points of interest in images. These operators have tended to use local radial symmetry as a measure of interest. This correlates well with psychophysical findings on fixation points of the human visual system. It has been observed that visual fixations tend to concentrate along lines of symmetry, (Locher and Nodine, 1987). Sela and Levine (1997) noted that the psychophysical findings of Kaufman and Richards (1969) corroborated this, placing the mean eye fixation points at the intersection of lines of symmetry on a number of simple 2D geometric figures. Figure 2.12(a) shows the results of Kaufman and Richards's study of mean spontaneous fixation positions for various small shapes, and Figure 2.12(b) shows the same shapes with their lines of symmetry annotated. It has also been observed that visual fixations are attracted to centers of mass of objects (Richards and Kaufman, 1969) and that these centers of mass are more readily determined for objects with multiple symmetry axes (Proffitt and Cutting, 1980).

One of the best known point of interest operators is the *generalized symmetry transform* (Reisfeld *et al.*, 1995). Figure 2.13 shows an example of the so-called *dark* and *radial* outputs of this transform. The transform highlights regions of high contrast and local radial symmetry and has been applied to detecting facial features (Reisfeld *et al.*, 1995; Intrator *et al.*, 1995; Reisfeld and Yeshurun, 1998).

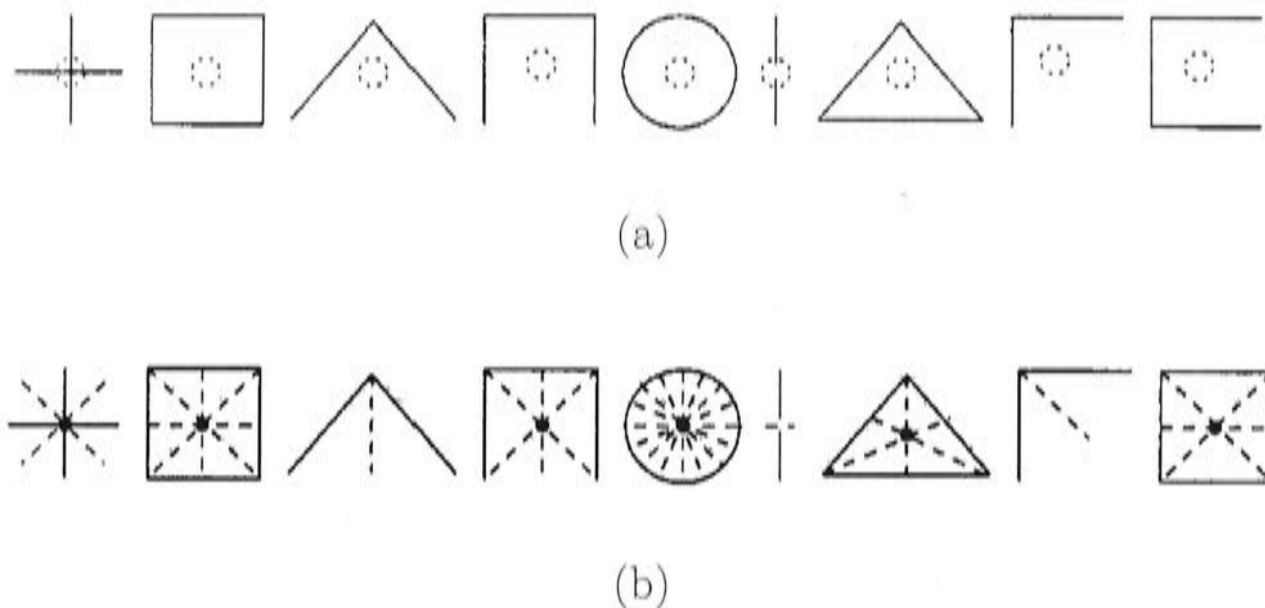


Figure 2.12: Modelling fixation tendencies, from Sela and Levine (1997). (a) Results of a study by Kaufman and Richards (1969) examining adult gaze fixation. The dotted circles indicate the location of mean spontaneous fixation. Each shape subtends two degrees of visual angle. (b) The same shapes with their lines of symmetry and their intersections displayed.



Figure 2.13: Examples from Reisfeld *et al.* (1995) showing (from left to right), a test image, and the *dark symmetry* and *radial symmetry* outputs of the Generalised Symmetry Transform.

It involves analyzing the gradient in a neighbourhood about each point. Within this neighbourhood the gradients at pairs of points symmetrically arranged about the central pixel are compared for evidence of radial symmetry, and a contribution to the symmetry measure of the central point is computed. The computational cost is high, being of order $O(KN^2)$, where K is the number of pixels in the image and N is the width of the neighbourhood. Whilst a realtime implementation has been attempted (Yamamoto *et al.*, 1994) it required a massive parallel computer architecture and was only able to achieve processing times of the order of seconds per frame.

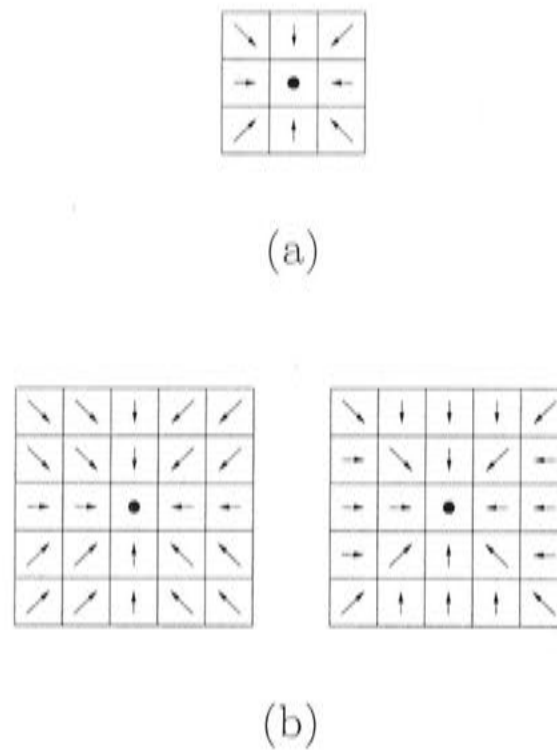


Figure 2.14: Gradient orientation masks used in Lin and Lin (1996) for detecting light blobs, (a) 3×3 mask, (b) 5×5 dual mask set.

Lin and Lin (1996) present a symmetry measure specifically for identifying facial features in images. They proposed a masking technique to evaluate radial symmetry based on gradient direction. Gradient directions are quantized into eight bins. The masks show which bin the local gradients should fall into for perfect radial symmetry about the center of the neighbourhood (for either a dark or light blob). Figure 2.14(a) shows the 3×3 gradient orientation mask for detecting light blobs (gradient pointing from dark to light). Dual-masks are used to accommodate for pixels where the acceptably radially-symmetric gradient orientations span two orientation bins, Figure 2.14(b) shows the dual mask set for a 5×5 neighbourhood. The radial symmetry at each pixel is determined by examining the discrepancy between the gradient orientations in the local neighbourhood and the orientation masks that represent perfect radial symmetry. The output of radially symmetric points from this comparison tends to be quite dense. In order to obtain points of radial symmetry useful for facial feature extraction two additional inhibitory processes are required: an edge map is used to eliminate all interest points which do not occur on edges, and regions of uniform gradient distribution are filtered out.

The computational cost of Lin and Lin's algorithm is stated as " $O(9K)$ " for an image of K pixels. However, within the definition of the algorithm the size of the local neighbourhood within which symmetry is determined is explicitly set to either 3×3 or 5×5 . Whilst the results for these values of $N = 3$ and $N = 5$

are good, no evidence is presented that this same level of performance will hold for larger neighbourhoods. In any case, extending this algorithm to measure symmetry in an $N \times N$ local neighbourhood results in a high computational cost of order $O(KN^2)$.

Sun *et al.* (1998) modify the symmetry transforms of Reisfeld *et al.* (1995) and Lin and Lin (1996) to obtain a symmetry measure which is combined with colour information to detect faces in images. An orientation mask is used similar to Lin and Lin (1996), together with a distance-weighting operator similar to Reisfeld *et al.* (1995), and the magnitude of the gradient is also taken into consideration. By using skin colour to initially identify potential face regions the scale of the symmetry operators can be chosen to suit the size of the skin region under consideration.

Sela and Levine (1997) present an attention operator based on psychophysical experiments of human gaze fixation. Interest points are defined as the intersection of lines of symmetry within an image. These are detected using a symmetry measure which determines the loci of centers of co-circular edges³ and requires the initial generation of an edge map. Edge orientations are quantized into a number of angular bins, and *inverted annular templates* are introduced to calculate the symmetry measure in a computationally efficient manner. Figure 2.15 shows one such template placed over edge point \mathbf{p} . Note that the direction of the gradient $\mathbf{g}(\mathbf{p})$ lies within the angular range of the template, and r_{min} and r_{max} specify the radial range of the template. Separate templates are required for different circle radii and gradient orientations. Convolution of one such template, of radius n and a particular angular range, with an image of edges, whose normals lie within this same angular range, generates an image showing the centers of circles of radius n tangential to these edges. This is repeated for each angular bin and each radius to form images of circle center locations. Co-circular points are then determined by examining common center points for circles of the same radius. The calculation of the final interest measure combines these points with orientation information of the corresponding co-circular tangents. This method can also be readily applied to log-polar images. The technique was shown to run in realtime on a network of parallel processors. The computational cost is of order $O(KBN)$ where B is the number of angular bins used (B is typically at least 8).

The approach of Sela and Levine bears some similarity to the circular Hough

³Two edges are said to be *co-circular* if there exists a circle to which both edges are tangent.

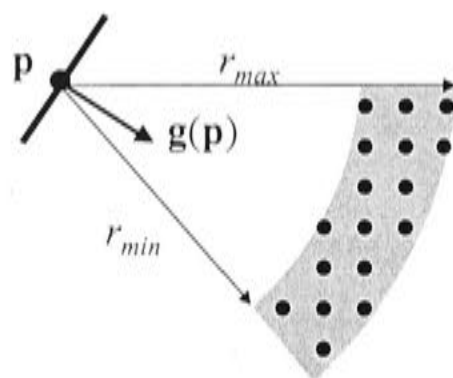


Figure 2.15: Inverted annular template as used by Sela and Levine (1997).

transform that is also used to find blobs in images. Duda and Hart (1972) showed how the Hough transform could be adapted to detect circles with an appropriate choice of parameter space. They required a three dimensional parameter space to represent the parameters a , b and c in the circle equation $(x - a)^2 + (y - b)^2 = c^2$. Kimme *et al.* (1975) noted that on a circle boundary the edge orientation points towards or away from the center of the circle, and used this to refine Duda and Hart's technique and reduce the density of points mapped into the parameter space. Minor and Sklansky (1981) further extended the use of edge orientation, introducing a *spoke filter* that plotted a line of points perpendicular to the edge direction (to the nearest 45 degrees) as shown in Figure 2.16. This allowed simultaneous detection of circles over a range of sizes (from r_{min} to r_{max} in Figure 2.16). An 8-bit code is generated for each point in the image, one bit for each of the eight 45 degree wide orientation bins. Each bit indicates whether a spoke filter of the appropriate orientation has plotted a point in a 3×3 neighbourhood about the point in question. Four discrete output levels are determined from the bit codes: all 8 bits positive, 7 bits positive, 6 adjacent bits positive, and all other cases. This technique was successfully used to detect blobs in infrared images. The computation required for an image of K pixels is of order $O(KBN)$ where B is the number of angular bins used (Minor and Sklansky (1981) used 8), and N is the number of radial bins.

Di Gesù and Valenti (1995a) present another method for measuring image symmetry called the *discrete symmetry transform*. This transform is based on the calculation of local axial moments, and has been applied to eye detection (Di Gesù and Valenti, 1995a), processing astronomical images (Di Gesù and Valenti, 1995b) and as an early vision process in a co-operative object recognition network (Chella *et al.*, 1999). The computational load of the transform is of the order $O(KBN)$ where K is the number of pixels in the image, N is the size of the local neigh-

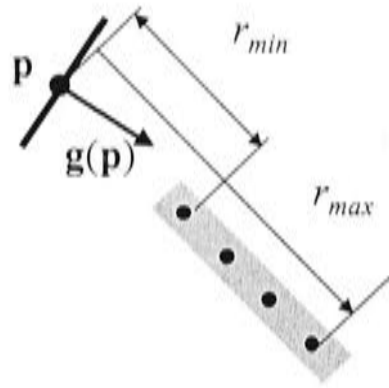


Figure 2.16: The spoke filter template proposed by Minor and Sklansky (1981).

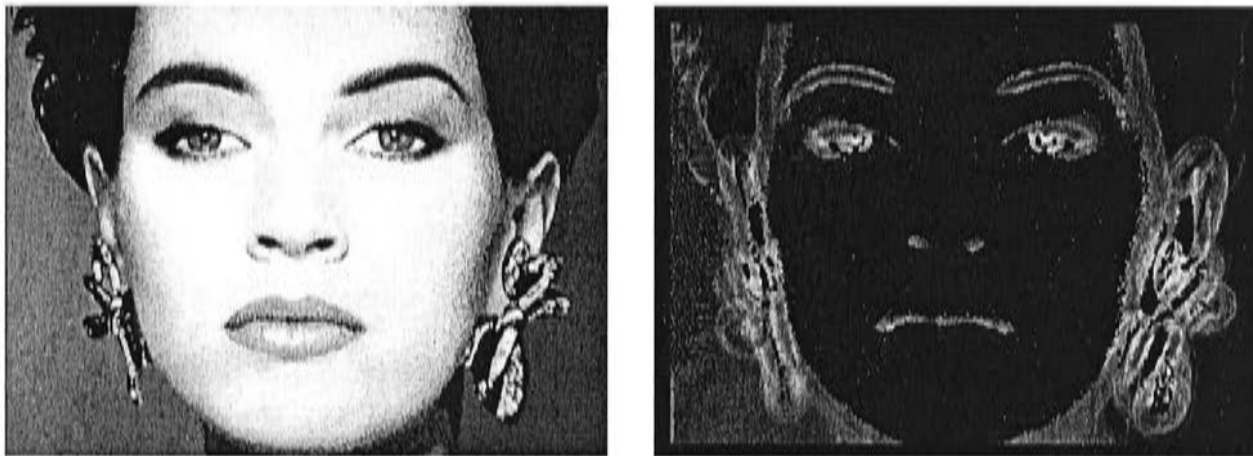


Figure 2.17: An example of the Discrete Symmetry Transform operating on a face image, from Di Gesù and Valenti (1995a).

bourhoods considered and B is the number of directions in which the moments are calculated. This load can be reduced by using a fast recursive method for calculating the moments (Alexeychuk *et al.*, 1997), giving a reduced computational order of $O(KB)$. Figure 2.17 shows an example of the transform being applied to detect the eyes in an image. Despite the strong highlighting of the eyes in this image, the transform tends to highlight regions of high texture in addition to radially symmetric points, note for instance the strong highlighting of the earrings in this example.

Kovesi (1997) presented a technique for determining local symmetry and asymmetry across an image from phase information. He notes that axes of symmetry occur at points where all frequency components are at either the maximum or minimum points in their cycles, and axes of *asymmetry* occur at the points where all the frequency components are at zero-crossings. Local frequency information is determined via convolution with quadrature log Gabor filters. These convolutions are performed for a full range of filter orientations and a number of scales,

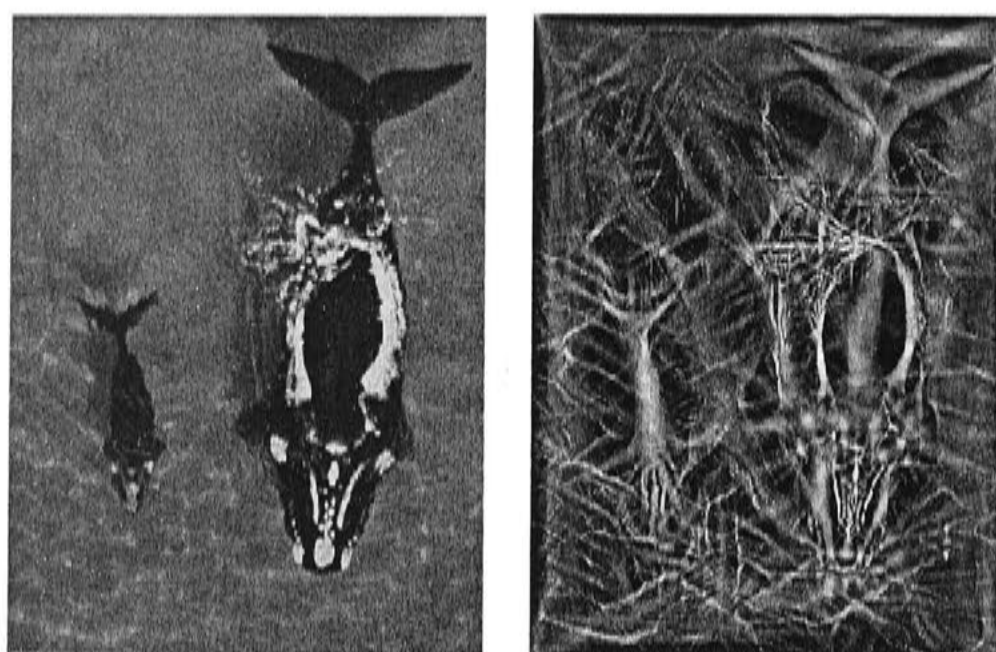


Figure 2.18: An example of symmetry from phase operating on a natural image, from Kovese (1997).

with each scale determining the response for a particular frequency bin. This technique is invariant to uniform changes in image intensity and as such is a truer measure of pure symmetry than other approaches which tend to measure a combination of symmetry and contrast. The computational cost of this method is high. Although the convolutions are efficiently performed in the frequency domain the computation required to transform the image between spatial and frequency domains is costly. This method is not intended as a point of interest operator. However, the resulting continuous symmetry measures it produces strongly corroborate the theory that points of interest lie on lines of symmetry. An example of the algorithm determining the symmetry across a natural image is shown in Figure 2.18. For a detailed discussion on image phase and its application see Kovese (1999a).

This section has demonstrated the suitability of radial symmetry-based feature detection for detecting facial features. There is no question that radial symmetry is a valuable cue. However, the best results for facial feature detection come from the generalized symmetry transform (Reisfeld and Yeshurun, 1998), and this transform is slow, computationally expensive to compute, and not well-suited to realtime applications. While some other methods provide more efficient alternative means of computing radial symmetry, the results obtained are not as useful for locating facial features. In Chapter 3 we present a new, computationally efficient method for determining radial symmetry that is able to produce results that rival those from the generalized symmetry transform whilst being fast enough to

operate in realtime.

2.2 Face Localisation

In Chapter 1 we identified three key steps to enabling a computer to see a face (see Figure 1.2), the first step is face localisation. Face localisation involves determining and tracking the location of a person's head in a complex dynamic scene. This is a challenging problem, especially if the system has to deal with changing lighting conditions, occlusions, and cluttered dynamic backgrounds.

Isard and Blake's famous condensation approach to contour tracking (Isard and Blake, 1996, 1998) tracks target's outlines using particle filtering and active contours. The outline of the target is parameterized using *B*-splines, and described as a point in state (parameter) space. Impressive results have been shown that illustrate how particle filter-based contour tracking methods can effectively deal with multiple hypotheses, occlusions and varying lighting conditions.

The particle filter approach to target localisation, also known as the condensation algorithm (Isard and Blake, 1996, 1998) and Monte Carlo localisation (Thrun, 2000), uses a large number of particles to "explore" the state space. Each particle represents a hypothesised target location in state space. Initially the particles are uniformly randomly distributed across the state space, and each subsequent frame the algorithm cycles through the steps illustrated in Figure 2.19:

1. Measure: The Probability Density Function (PDF) is measured at (and only at) each particle location. Thus a probability measure is assigned to each particle indicating the likelihood that that particle is the target.
2. Resample particles: The particles are re-sampled with replacement, such that the probability of choosing a particular particle is equal to the probability assigned to that particle.
3. Deterministic drift: Particles are moved according to a deterministic motion model.
4. Diffuse particles: Particles are moved a small distance in state space under Brownian motion.

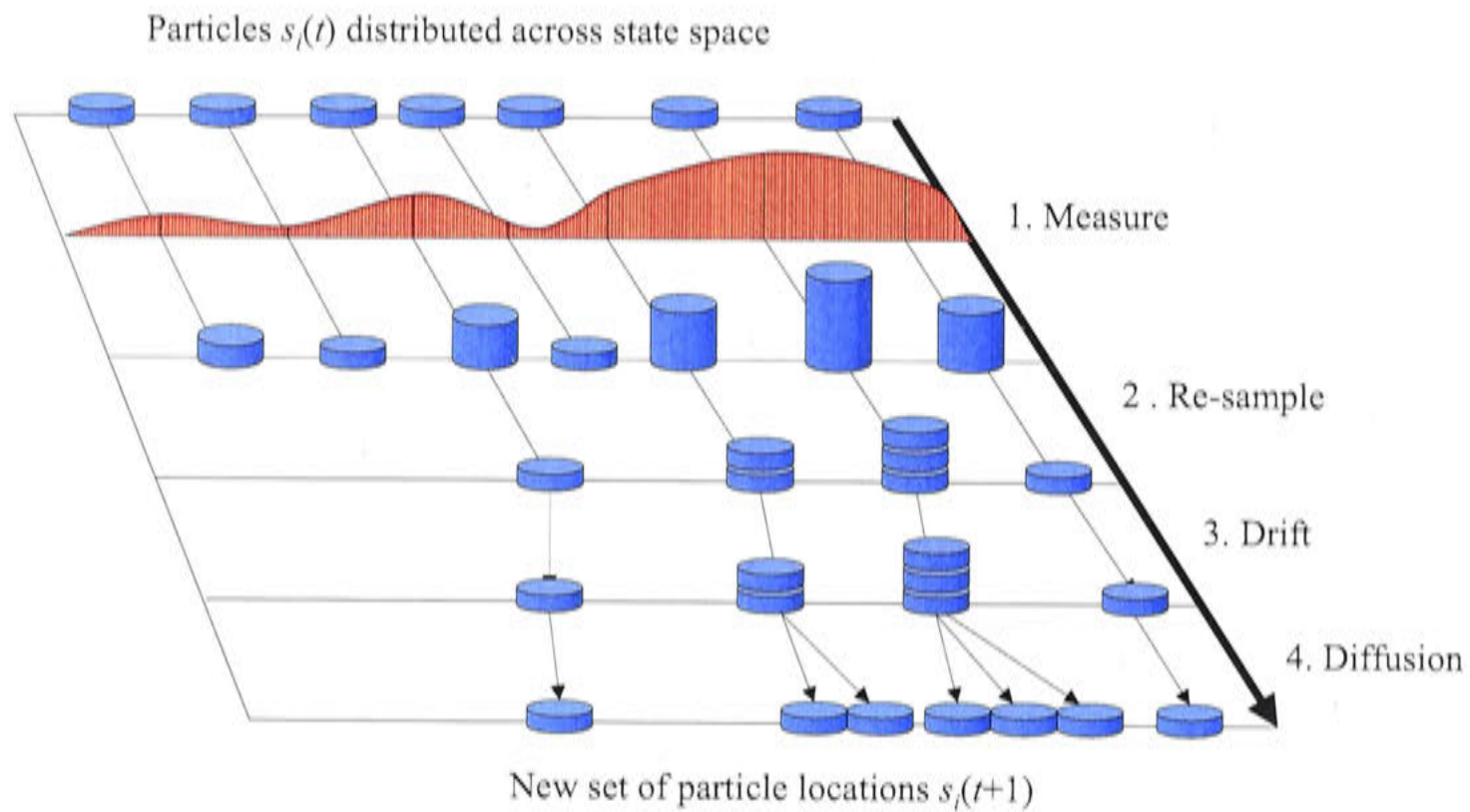


Figure 2.19: Evolution of particles over a single time-step. The unknown PDF is measured only at the particle locations, particles are then re-sampled with replacement, and drift and diffusion are applied to evolve the particles to their new locations.

Note that any dynamics can be used in place of steps 3 and 4, but the standard approach is to apply drift and diffusion.

This cyclic process results in particles congregating in regions of high probability and dispersing from other regions, thus the particle density indicates the most likely target states. Furthermore the high density of particles in these “target-like” regions means that these regions are effectively searched at a higher resolution than other more sparsely populated regions of state space.

The hypothesis-verification approach used by particle filters is a powerful method for locating targets in state space. It is especially attractive as it does not require the probability density function to be calculated across the entire state space, but only at the particle locations. Using this approach to locate a target in an image does not require searching across the entire image in the usual manner — as is done with template matching across a region, for instance — instead we need only verify targets at hypothesised locations. The challenge is then to ensure that the hypotheses end up finding the target.

There are several things that can be done to maximise the likelihood of the hypotheses converging on the target. Firstly, there is the design of the particle filter: using a sufficient number of particles, appropriate diffusion parameters, and

a valid motion model to approximate the target's motion and facilitate calculation of the deterministic drift. Secondly there is the choice of cues used to measure the PDF at each hypothesis location. The process benefits greatly from cues whose responses increase steadily in the vicinity of the target location, rather than cues (such as normalised cross correlation) that give a high response only at the target location and noise elsewhere. By using cues that give a high response when close to (as well as) the target location the particle filter is able to propagate hypotheses that are close to likely target locations, and thus increase the resolution of the search at these locations, without relying on a hypothesis being located precisely at the target location in order to generate a high response.

MacCormick and Blake (1998) describe a generic object localisation technique designed to initialise a contour tracker such as the one proposed by Isard and Blake (1996, 1998). Their system is able to locate a target in a cluttered environment, requires no knowledge of the background, and is robust to lighting changes. Rather than searching the entire image, a large number of hypothesis target locations are considered (MacCormick and Blake use 1,000). Each one of these is evaluated using Bayesian probability theory to quantify whether it is more "target-like" or "clutter-like". Hypotheses are chosen based on a prior statistical density describing the likelihood of a target occurring at a given position in state space. This density is determined from a training sequence of the target exhibiting typical behaviour, in which the target is tracked using a manually initialised contour tracker. The frequency of different state space configurations observed in this training sequence is used to build the density describing the likelihood of a given hypothesis configuration occurring.

Using multiple visual cues is known to improve the robustness and overall performance of target localisation systems. A number of researchers have utilised multiple cues to detect and track people in scenes, however, there have been few attempts to develop a system that considers the allocation of finite computational resources amongst the available cues, the notable exception being Crowley and Berard (1997).

Crowley and Berard (1997) used multiple visual processes: blink detection, colour histogram matching, and correlation tracking, together with sound localisation, to detect and track faces in video for video compression and transmission purposes. Each cue is converted to a state vector containing four elements: the x and y coordinates of the centre of the face, and the face height and width. A confidence

measure and covariance matrix are estimated by examining the state vectors of all the cues, and used to combine the state vectors to give the final result. The advantage of this approach is the extremely compact form in which the state vectors represent information. The disadvantage is that it only allows one face target to be reported by each cue. Apart from the inability of such a system to deal with multiple faces, it only allows each cue to report a single target and thus throws away any additional information the cue may provide. For instance, if there are two regions of skin-like colour we would prefer a system to report the presence of both regions and allow the additional cues to determine which is a face, rather than returning a single result, namely the centre of gravity of the two regions.

Kim and Kim (2000) combine skin colour, motion and depth information for face detection. Initially depth information is used to segment objects from the background, then the AND operator is used to combine the information from the colour and motion cues. This is the simplest way of combining information, and it will reduce the number of false positives. However, it is only suitable for cues in binary form, and although any set of continuous cues can easily be converted to binary, doing so throws away a great deal of information which is useful for determining the confidence and reliability of the cue's performance. As such, combining cues with the AND operator is only suitable when the performance level of each cue is known, and is undesirable for a system which must be robust to varying operating conditions.

Darrell *et al.* (2000) integrate stereo, colour, and face detection to track a person in a crowded scene in realtime. A stereo depth map is used to isolate silhouettes of the subjects, and a skin colour cue identifies and tracks likely body parts within these silhouettes. Face pattern detection is applied to discriminate the face from other detected skin-coloured regions. The system tracks users over various time scales and is able to recognise a user who returns minutes — or even days — later. Statistics gathered from all three modalities are used to recognise users who reappear after becoming occluded or leaving the scene. This system demonstrates the advantage of fusing multiple cues for robustness and speed: using the simple but efficient depth and colour cues to localise targets in realtime before following through with the slower, yet more precise, face detection module. The disadvantage to applying cues in a serial manner such as this is the implicit requirement that the initial cues must not miss the target. This problem can be minimized, however, by accepting an increased number of false positives

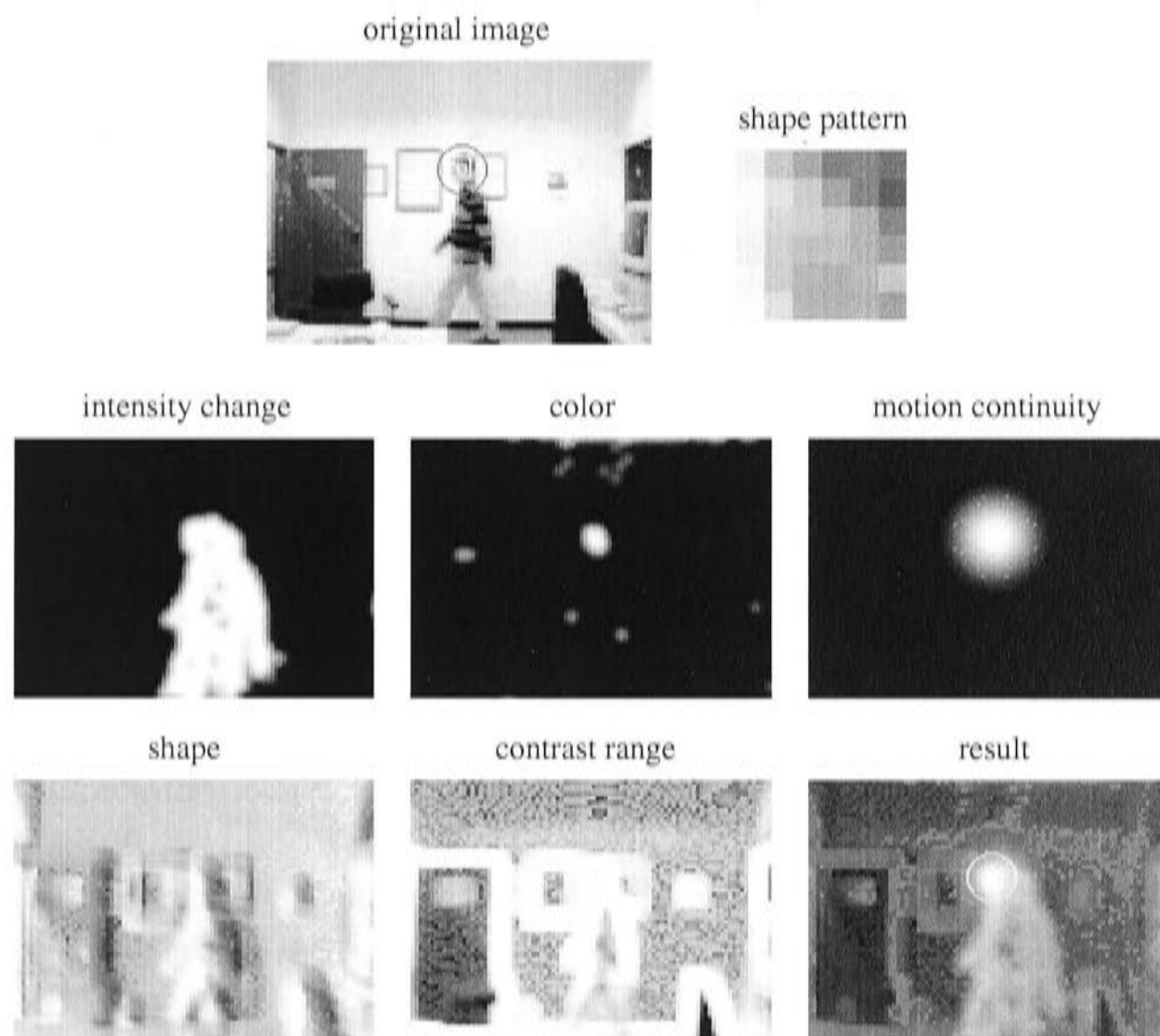


Figure 2.20: Cues operating in Triesch and von der Malsburg's system, from Triesch and von der Malsburg (2000).

from the initial cues.

Triesch and von der Malsburg (2000) present a system suitable for combining an unlimited number of cues. The system is demonstrated using contrast, colour, shape, and two motion cues (intensity change and a predictive motion model), to track a person's head. The results of these cues together with an input image and target shape model are shown in Figure 2.20.

For each (i^{th}) cue and (k^{th}) image frame the following quantities are determined:

- an image of probabilities $\mathbf{A}_i[k]$ describing the probability a given pixel is part of a face (as shown for each cue in Figure 2.20),
- a quality measure $q_i[k]$ describing how accurate the sensor was in determining the final result in the previous image frame, and
- a reliability measure $r_i[k]$, which is effectively a running average of the



Figure 2.21: Triesch and von der Malsburg’s system tracking a person’s head in an image sequence, from Triesch and von der Malsburg (2000). These frames are taken across a 5 second period and show robustness to changing lighting conditions.

quality measure $q_i[k]$.

The final result is given by the weighted sum $\sum_i r_i[k] \mathbf{A}_i[k]$. The $\mathbf{A}_i[k]$ image is generated by comparing the i^{th} sensor’s information with a prototype $\mathbf{P}_i[k]$ which describes the target (a face) with respect to that sensor. These prototypes are updated dynamically as a running average of the sensor’s output at the target locations in previous frames.

The results of this system (see for example Figure 2.21) are impressive and demonstrate how combining multiple cues increases the robustness of a tracking system. This system was an inspiration for our work, which, however differs in several aspects. Firstly, Triesch and von der Malsburg’s system is primarily a tracking system rather than a localisation system. The principal requirement of a localisation system is to ensure that the object found fits the generic requirements of the target (in this case a face), whereas a tracking system is primarily concerned with locating the same object repeatedly over a series of frames. The use of running averages to adapt the sensor fusion suite to the target identified in previous frames is well suited for tracking applications, but is less appropriate for a target localisation system, as it is undesirable to dynamically change a localisation system’s perception of what the target should look like, lest the system be distracted from the true target. Secondly, we require systems to localise a target in 3D, whereas this system operates in 2D, and with fixed sized prototypes it cannot deal with close-up or far-away targets. Finally, when determining the usage of different cues we wish to take into account not only the tracking performance, but also the computational requirement of each cue.

Recent work by Soto and Khosla (2001) presents a system based on *intelligent agents* that adaptively combines multi-dimensional information sources (*agents*) to estimate the state of a target. A particle filter is used to track the target’s

state, and metrics are used to quantify the performance of the agents. Initial results for person tracking in 2D show a good deal of promise for a particle filter based approach.

This section has discussed a number of systems that have been developed to address the problem of robustly localising a face (or other target) in a complex environment. The particle filtering approach popularised by Isard and Blake (1996, 1998) offers a solid framework for locating and tracking targets, and as Soto and Khosla (2001) demonstrated it is well suited for use in a multi-cue system. There is no question that multiple cues allow for more robust estimates, however, calculating more cues requires more CPU time and can quickly reach the limits imposed by a realtime system. Few researchers have considered the problem of controlling the allocation of computational resources between cues, in order to allow more effective and efficient cues to operate at the expense of those that are slower or not performing as well.

The face localisation system that we present in Chapter 4 aims to meld the strongest elements of the systems discussed in this section. A particle filter is used to maintain multiple hypotheses of the target's location, and multiple visual cues will be applied to test hypotheses. Finite computational resources will be allocated across the cues, taking into account the cue's expected utility and resource requirement. Our system accommodates for cues running at different frequencies, allowing cues performing less well to be run slowly in the background for added robustness with minimal additional computation.

2.3 Face Registration

After face localisation the second step towards enabling a computer to see a face is *face registration* (see Figure 1.2). We use the term *face registration* to refer to the process of registering the locations of facial features and verifying that the image region in question does indeed contain a face, see for example Figure 2.22. This is a specialisation of the general problem of face detection that typically involves determining the locations of faces in an image, and may or may not be extended to locating facial features. Over the last decade the problem of face detection in images has received a growing amount of attention from researchers in commercial and academic institutions alike. It is widely recognised that face detection is the first step towards face recognition and a myriad of other human



Figure 2.22: Face registration. (a) An image containing a face. (b) Presence of face verified and facial features detected. This example is from our system described in Chapter 5

computer interaction tasks. Recent survey papers by Yang *et al.* (2002) and Hjelmås and Low (2001) provide an excellent overview of the field and reveal the quantity and diversity of research that has gone into detecting faces and facial features.

In 1973 Kanade pioneered the use of integral projection to locate the boundaries of a face. Since then integral projection and variations thereof have been used to detect facial features in a number of applications (Kotropoulos and Pitas, 1997; Katahara and Aoki, 1999; Chuang *et al.*, 2000). Integral projection involves projecting the values of image pixels onto an axis. The integral projections of an image \mathbf{I} onto the x (horizontal) and y (vertical) axes are respectively given by

$$\mathbf{p}_x(x) = \sum_y \mathbf{I}(x, y)$$

and

$$\mathbf{p}_y(y) = \sum_x \mathbf{I}(x, y).$$

Taking the integral projection of an image onto the horizontal (x) axis amounts to summing the pixel values down each column, and results in a vector that is literally the projection of the integral of each column onto the horizontal axis. Likewise, integral projection onto the vertical axis is a vector containing the sum of pixel values in each row of the image. It is feasible to perform integral projection onto any axis, but in practice vertical and horizontal integral projection are most commonly used.

The integral projection method is simple and fast. It is useful for detecting features whose intensities stand out from the background, especially those with

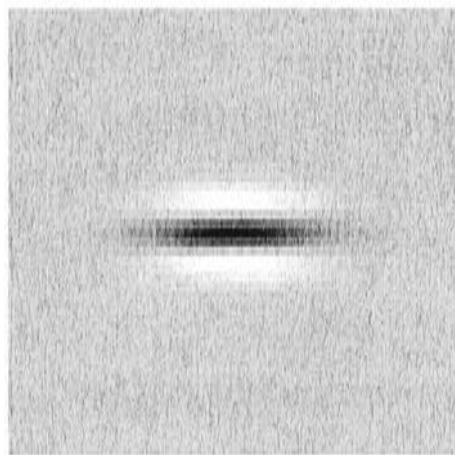


Figure 2.23: The kernel used by Yow and Cipolla (1997) it is a second derivative of a Gaussian in one direction, Gaussian in the orthogonal direction, and elongated with an aspect ratio of 3:1 (Figure from Yow and Cipolla (1995)).

a strong horizontal or vertical aspect. As such it is well suited to detecting facial features. The intensities of facial features generally stand out strongly against the skin of the face, and vertical integral projection is especially well-suited for upright faces owing to the dominant horizontal aspect of most facial features.

The main problems when applying this method to detect facial features are: segmenting the facial region from the image to avoid background interference, ensuring that the desired features stand out to the exclusion of everything else, and requiring the face to be in an upright position. If these problems are addressed then integral projection provides an excellent way of locating facial features.

Spatial filtering can be used to enhance and identify facial feature candidates (Graf *et al.*, 1995; Yow and Cipolla, 1997). In this process the intensity image is typically smoothed, then convolved with specially chosen kernels to extract the facial features, for example, long thin kernels are used to detect eyes. Yow and Cipolla (1997) use the elongated Gaussian-based kernel shown in Figure 2.23, while Graf *et al.* (1995) use rectangular kernels and subtract the result from the original image. The latter claimed this approach was adequate for separating the eyes, mouth, and tip of the nose from the cheeks, forehead, and chin, and went on to use a morphological approach to enhance the image at points identified by the filtering. Spatial filtering is orientation and scale dependent, however, a small deviation of the target from the intended orientation and scale can be tolerated.

Edge information is useful for detecting and verifying features. After identifying possible facial features using spatial filtering, Yow and Cipolla (1997) discard any feature which does not have parallel edges bounding it from above and below. If the face is assumed to be upright this simply involves looking for pairs of

horizontal lines, if the face orientation is unknown it is a slightly more complicated operation. Lin and Lin (1996) note that artistic sketches of human faces can faithfully represent subjects by using simple sketching lines corresponding to edges of the features. They initially employ a region of interest detector to identify potential facial feature candidates (see Section 2.1.5) and then disregard all such features which are not coincident with edges in the image.

Radial symmetry can be used to detect facial features and is especially well suited to detecting eyes. A number of radial symmetry-based feature detectors are discussed in Section 2.1.5. The best results are from Reisfeld *et al.*'s generalised symmetry, however, this method is very computationally intensive. While some of the alternative methods offer more efficient computation their performance as facial feature detectors is not as promising.

Blink detection has been shown to be effective for locating eyes (Crowley and Coutaz, 1995). Blinking is a distinctive motion, especially as both eyes blink at once, so the movement occurs at two distinct locations simultaneously, and can be easily distinguished from most other movement in an image sequence. In 1995 Crowley and Coutaz presented a face localisation algorithm relying solely on blink detection. Later this system was augmented to utilise other sensing modalities for face detection (Crowley and Berard, 1997) (see Section 2.2). Blink detection is simple, computationally cheap, and reliable, but it does require waiting for the subject to blink.

Turk and Pentland (1991) used the method of principal component analysis to form a reduced basis of eigenvectors, dubbed "eigenfaces", from a large training set of aligned and equisized face images. Principal component analysis enables a small number of eigenfaces to be extracted that form a basis that spans almost the entire training set. Furthermore, the projection of *any* image onto this basis of eigenfaces will be a linear combination of the eigenfaces, so is restricted to have some sort of face-like appearance. The process of quantifying how face-like a test image is simply involves comparing the test image with its projection onto the basis of eigenfaces. The projection can be efficiently determined and is simply the linear combination of eigenfaces with the coefficient of each eigenface given by its scalar vector product with the test image. Turk and Pentland extended the concept beyond eigenfaces to eigenfeatures applying principal component analysis on individual facial features, and found that recognising faces using eigenfeatures was more robust than simply using eigenfaces alone. The eigenfeature approach

can be used to search for facial features in images, and provides a compact way of comparing a test image with a large population of similar images. However, it becomes very computationally expensive when used to perform an exhaustive correlation-style search for a target.

Colour information can be used for detecting facial features. Oliver *et al.* (1997) uses colour to locate the mouth. Varchmin *et al.* (1997) noted that nostrils often appear as bright spots in the red colour channel. Colour gradient can also provide useful information, potentially allowing a system to discriminate between white features (such as the eye whites and teeth), points of reflection off shiny surfaces (such as the eyeball or bright metal jewelry) and reflection off less shiny surfaces such as skin.

In summary, integral projection is a simple yet powerful technique for detecting isolated features whose intensities stand out clearly from the background. It is necessary to preprocess face images to prepare them for integral projection, the face must be aligned so it appears upright in the image, it is also beneficial to enhance the features so that they distinctly stand out within the face region. Spatial filtering methods have been shown to enhance and detect facial features using smoothing and specially designed kernels aligned with the features. The usefulness of radial symmetry for detecting facial features has been demonstrated, however, the methods that return the best results are computationally intensive; a faster, more efficient method is needed to make this a viable option. Finally blink detection can provide a simple, efficient and reliable method for identification of eye locations, the drawback is that it is necessary to wait for the subject to blink.

In Chapter 5 we present a face detection system that uses blink detection to initially localise the eye and face location, and apply filtering and radial symmetry detection to enhance facial features. Finally, feature locations are pin-pointed using integral projection.

2.4 Face Tracking

The final step (as depicted in Figure 1.2) to enable a computer to see a face is face tracking. Face tracking involves both tracking the pose of the head in 3D space, and the location of facial features. Some facial features, such as the eyes and nose are rigidly attached to the head and their motion can be directly linked to the

head pose. Other features, such as the mouth and eyebrows, are deformable, and their location is a function of both the head pose and their own deformation. We will consider tracking both rigid and deformable facial features, and accordingly this section is divided into two parts. The first considers tracking rigid facial features in order to determine the head pose, and the second looks at tracking deformable facial features. Particular emphasis is placed on tracking the lips and mouth contour owing to the relevance of mouth-shape information for Human Computer Interaction.

2.4.1 Tracking Rigid Facial Features

Our primary interest in tracking rigid facial features is to determine the head pose, that is, the location and orientation of the head in 3D space. The head can be modelled as a rigid body with a number of features rigidly attached, these features include the eye sockets, eyes, nose and hairline. By tracking the locations of features rigidly attached to the head it is feasible to track the pose of the head. A reference frame is attached to the head, and the pose of the head is defined by a six parameter vector $(x, y, z, \theta_x, \theta_y, \theta_z)$ specifying the Cartesian co-ordinates and rotation of the head reference frame with respect to a predefined world coordinate system. Figure 2.24 shows a schematic of a head with reference frame attached showing the pose of the head reference frame in the world coordinate system.

Estimating the 3D pose of a rigid object requires determining the six parameter state vector: $(x, y, z, \theta_x, \theta_y, \theta_z)$ specifying the Cartesian co-ordinates and rotation with respect to a predefined reference frame.

Lowe's object tracking algorithm (Lowe, 1991) presents a model-based approach to determining the pose of a known 3D object. Model-based vision uses prior knowledge of the structure being observed to infer additional information than is otherwise evident from an image. When a 3D object is viewed in an image the locations of its features are a non-linear function of the pose of the object relative to the camera. Given an initial guess of the pose, a least squares solution can be achieved iteratively by applying Newton's method to locally linearize the problem. Lowe augments this minimization in order to obtain stable approximate solutions in the presence of noise. This is achieved by incorporating a model of the range of uncertainty in each parameter, together with estimates of the standard deviation of the image measurements, into the minimization procedure. On top of this Lowe

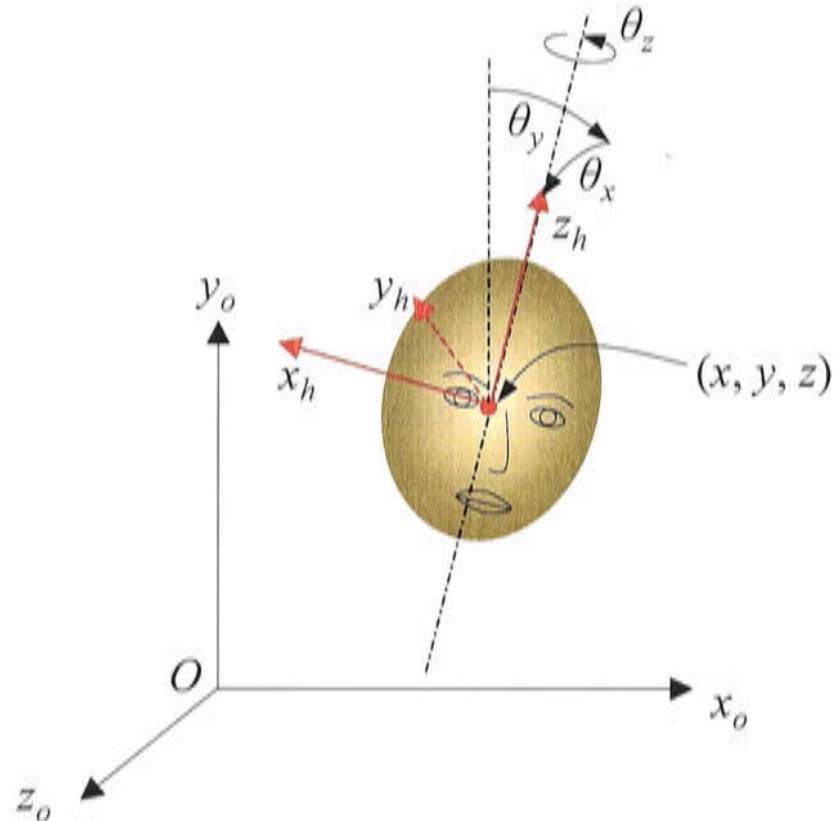


Figure 2.24: 3D pose of a head. Head reference frame shown in orange, and the pose $(x, y, z, \theta_x, \theta_y, \theta_z)$ with respect to the world coordinate frame O indicated.

applies the Levenberg-Marquardt method to ensure the solution converges to a local minima. Lowe demonstrated that this method could efficiently track the pose of known 3D objects with complex structures and provide reliable results. This algorithm provides an attractive means of tracking the pose of a known 3D object in a monocular image sequence.

Azarbayejani *et al.* (1993) implemented a Kalman filter to track the head pose using an approach similar to that adopted by Clark and Kokuer (1992) and Reiniers *et al.* (1992) for calculating the orientations of objects. Azarbayejani *et al.* extract feature templates in an initial image, and use normalised cross correlation to locate these features in subsequent image frames. The head pose is iteratively determined using an extended Kalman filter with an 18-dimensional state vector containing a concatenation of the six 3D pose parameters and their first and second derivatives. Measurement variances are determined from the correlation values obtained from the feature templates. Despite the non-linear relationship between the observed 2D feature locations and the pose parameters, the local linearization employed by the extended Kalman filter was shown to provide suitable tracking results. Azarbayejani and Pentland (1995) later extended this method to recover not only the 3D pose of the head (or other 3D object) but also the 3D structure of the object itself, along with the focal length of the camera.

Gee and Cipolla (1994) used four facial features, namely the pupils and mouth corners, to track the head pose. These features were assumed to lie in a plane, and two vectors are determined: one joining the eyes, and one joining the mid-point of the eyes with the mid-point of the mouth corners. From these vectors a third vector is calculated normal to the face that described the head pose. Maurer and von der Malsburg (1996) also tracked facial features and assumed they lay in a plane, however, they used more features than Gee and Cipolla. The head pose was determined by solving the resulting over-constrained system using least squares. Shakunaga *et al.* (1998) used a similar approach but did not assume the features lay in a plane. They solved for the pose under orthographic projection and could cope with an arbitrary number of features.

Xu and Akatsuka (1998) track the head pose by reconstructing the 3D locations of facial features using stereo. The pupils and mouth corners are tracked using stereo and their 3D locations determined. The pose is determined as the normal to the plane defined by the pupils and a mouth corner.

Matsumoto and Zelinsky (2000) also made use of stereo for their Karman filter-based solution to the head tracking problem. This system used calibrated stereo cameras and was able to run in realtime and determine the head pose with higher accuracy than the method proposed by Azarbajani *et al.*. Recently this system has been evolved into the commercial FaceLab system by Seeing Machines⁴. It requires no markers or special make-up to be worn and runs on a standard PC. The software consists of three key parts, 3D Facial Model Acquisition, Face Acquisition, and 3D Face Tracking.

The Face Model Acquisition module builds a model of the subject's face off-line. The face model consists of up to 32 features ($\mathbf{T}_i, i = 0, 1, 2, \dots$) corresponding to a set of 3D model points ($\mathbf{m}_i, i = 0, 1, 2, \dots$) in the head reference frame. The head frame is placed between the eyes and oriented as shown in Figure 2.24.

The system starts operation in Face Acquisition mode where it attempts to find an initial lock on the face in the image stream. During this phase a template constructed from the edge map of the entire central region of the face is searched for. This template is automatically extracted during the model acquisition phase where the position of the face in the image is known. Normalised correlation matching is used both here and during tracking to make this process robust to changes in lighting conditions.

⁴<http://www.seeingmachines.com>

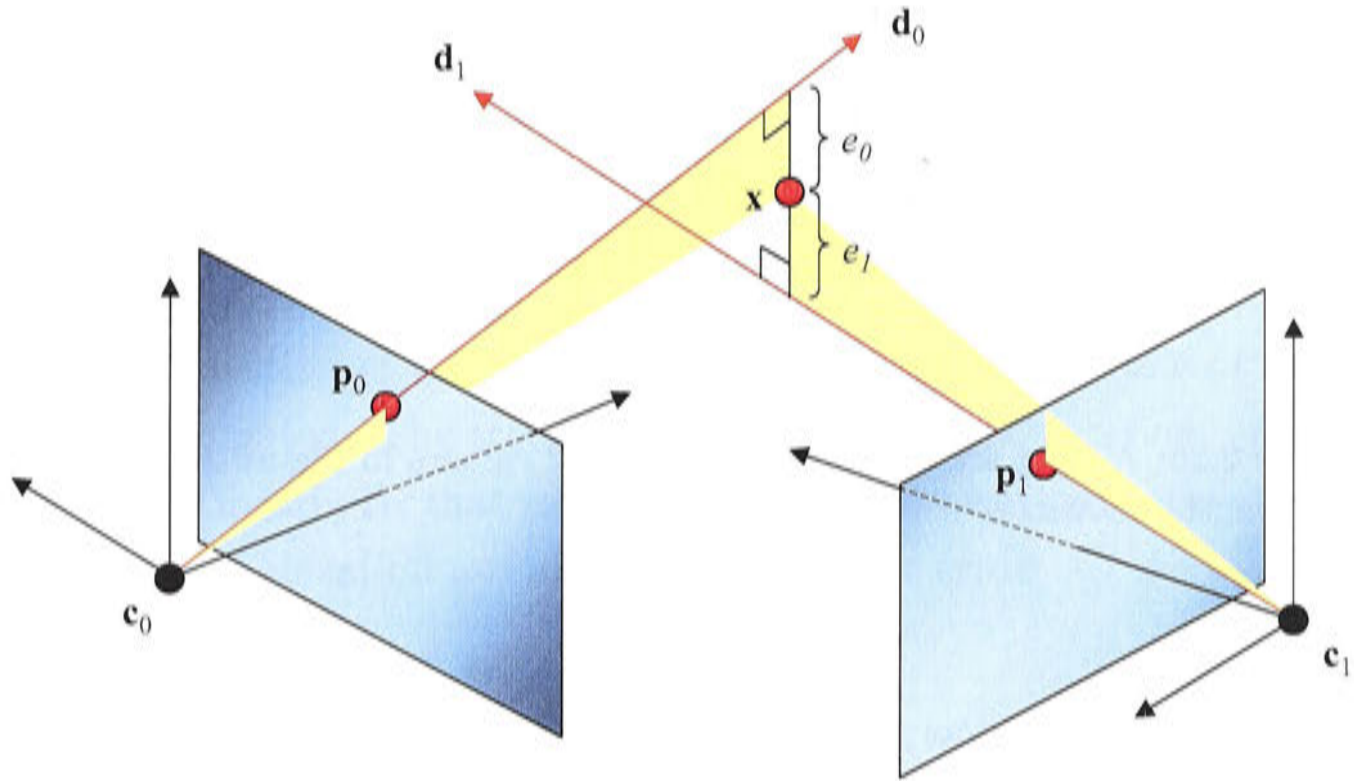


Figure 2.25: 3D reconstruction from stereo images.

When a match is found with a correlation above a preset value, the approximate positions of the features \mathbf{T}_i are identified based on their known offsets from the centre of the face (again calculated during model acquisition). Tracking is performed using the templates \mathbf{T}_i obtained during model acquisition. These are correlated with the current stereo view in the input stream and their 3D positions are calculated using linear triangulation. This technique is described below (for more detail the reader is referred to Trucco and Verri (1998)).

Ideally the 3D rays projected from the camera centres through the observed feature points on the image plane will intersect, defining the 3D location of the feature point. However, in general, owing to small errors in feature locations or camera parameters, the rays will not meet. This situation is illustrated in Figure 2.25. Linear triangulation proceeds to determine the location for the 3D point \mathbf{x} that minimizes the distances e_0 and e_1 . More specifically for n cameras linear triangulation minimizes E in

$$E = \sum_{i=0}^n e_i^2 \quad (2.6)$$

Returning our attention to the Figure 2.25, the distances, e_0 and e_1 can be expressed in terms of \mathbf{x} by observing that they are side lengths of right angle triangles (indicated in yellow). Considering each of these triangles separately, the side

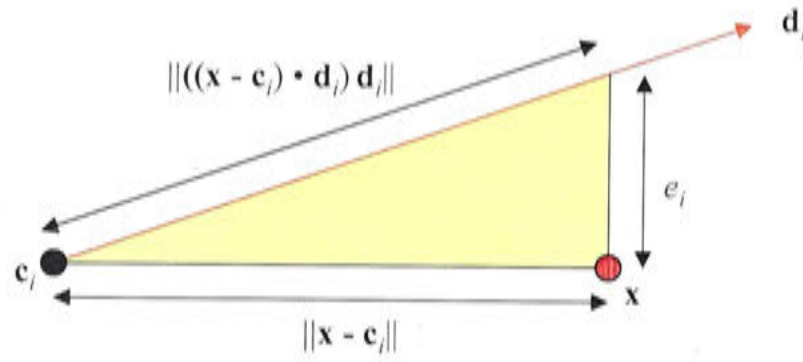


Figure 2.26: A right angle triangle from the i^{th} camera in Figure 2.25 with all side lengths shown.

lengths can be expressed as shown in Figure 2.26, and thus e_i^2 can be written as

$$e_i^2 = \|((\mathbf{x} - \mathbf{c}_i) \cdot \mathbf{d}_i) \mathbf{d}_i\|^2 - \|\mathbf{x} - \mathbf{c}_i\|^2$$

where \mathbf{d}_i is a unit vector along the optical axis of the i^{th} camera. For the case of two cameras Equation 2.6 can be expanded to

$$E = \|((\mathbf{x} - \mathbf{c}_0) \cdot \mathbf{d}_0) \mathbf{d}_0\|^2 - \|\mathbf{x} - \mathbf{c}_0\|^2 + \|((\mathbf{x} - \mathbf{c}_1) \cdot \mathbf{d}_1) \mathbf{d}_1\|^2 - \|\mathbf{x} - \mathbf{c}_1\|^2 \quad (2.7)$$

Setting the partial derivatives of this equation with respect to the elements of \mathbf{x} to zero gives a system of linear equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where \mathbf{A} and \mathbf{b} are

$$\mathbf{A} = \begin{pmatrix} (d_{0x}^2 - 1) + (d_{1x}^2 - 1) & d_{0x}d_{0y} + d_{1x}d_{1y} & d_{0x}d_{0z} + d_{1x}d_{1z} \\ d_{0x}d_{0y} + d_{1x}d_{1y} & (d_{0y}^2 - 1) + (d_{1y}^2 - 1) & d_{0y}d_{0z} + d_{1y}d_{1z} \\ d_{0x}d_{0z} + d_{1x}d_{1z} & d_{0y}d_{0z} + d_{1y}d_{1z} & (d_{0z}^2 - 1) + (d_{1z}^2 - 1) \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} d_{0x}\mathbf{c}_0 \cdot \mathbf{d}_0 - c_{0x} + d_{1x}\mathbf{c}_1 \cdot \mathbf{d}_1 - c_{1x} \\ d_{0y}\mathbf{c}_0 \cdot \mathbf{d}_0 - c_{0y} + d_{1y}\mathbf{c}_1 \cdot \mathbf{d}_1 - c_{1y} \\ d_{0z}\mathbf{c}_0 \cdot \mathbf{d}_0 - c_{0z} + d_{1z}\mathbf{c}_1 \cdot \mathbf{d}_1 - c_{1z} \end{pmatrix}$$

and $d_{ix,y,z}$ and $c_{ix,y,z}$ are the elements of \mathbf{d}_i and \mathbf{c}_i respectively. These equations can be solved for \mathbf{x} ,

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

giving the 3D location of the point.

A more sophisticated alternative to linear triangulation is Hartley and Sturm's optimal triangulation method that minimizes the error observed in the images subject to the epipolar constraint (Hartley and Sturm, 1995; Hartley and Zisserman, 2000). However, the linear triangular method detailed above provides suitable performance for the 3D head tracking system.

Once the 3D position of the features are determined an estimate of the pose of the head is computed. The translation vector \mathbf{t} , and the rotation, encapsulated in the rotation matrix \mathbf{R} , that together describe the head pose are estimated via least squares minimization as follows. Minimize the error

$$E = \sum_{i=1}^n w_i \|\mathbf{x}_i - \mathbf{R}\mathbf{m}_i - \mathbf{t}\|^2 \quad (2.8)$$

where \mathbf{x}_i is the measured 3D feature location, \mathbf{m}_i is the 3D model point, and w_i is the weighting factor for the i^{th} feature. The value of the weighting factor is set to the correlation value obtained for the associated feature in the template tracking step. This applies a more dominant weighting to features that returned higher correlation values making the system more robust to mismatched features.

The translation \mathbf{t} is determined by differentiating Equation 2.8 and setting the result to zero, yielding,

$$\mathbf{t} = \bar{\mathbf{x}} - \mathbf{R}\bar{\mathbf{m}} \quad (2.9)$$

where

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^n w_i \mathbf{x}_i}{\sum_{i=1}^n w_i}$$

and

$$\bar{\mathbf{m}} = \frac{\sum_{i=1}^n w_i \mathbf{m}_i}{\sum_{i=1}^n w_i}$$

are weighted averages of the measured features locations and the model points respectively.

Substituting \mathbf{t} from Equation 2.9 into Equation 2.8 and ignoring all terms that are not dependent on \mathbf{R} gives us

$$E' = 2 \sum_{i=1}^n w_i (\mathbf{x}_i - \bar{\mathbf{x}})^\top \mathbf{R} (\bar{\mathbf{m}} - \mathbf{m}_i)$$

Using the quaternion representation for a rotation matrix \mathbf{R} can be written as

$$\mathbf{R} = \begin{pmatrix} a^2 + b^2 - c^2 - d^2 & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & a^2 - b^2 + c^2 - d^2 & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ab) & a^2 - b^2 - c^2 + d^2 \end{pmatrix} \quad (2.10)$$

where a , b , c and d are real numbers and $a^2 + b^2 + c^2 + d^2 = 1$.

The method of Lagrange multipliers can then be used to minimize E' as follows. Define

$$E'' = 2 \sum_{i=1}^n w_i (\mathbf{x}_i - \bar{\mathbf{x}})^\top \mathbf{R} (\bar{\mathbf{m}} - \mathbf{m}_i) + \lambda (a^2 + b^2 + c^2 + d^2 - 1)$$

Determine the partial derivatives of E'' with respect to a , b , c and d , and set these to zero. This gives the following four linear equations,

$$\sum_{i=1}^n w_i (\mathbf{x}_i - \bar{\mathbf{x}})^\top \begin{pmatrix} a & -d & c \\ d & a & -b \\ -c & b & a \end{pmatrix} (\bar{\mathbf{m}} - \mathbf{m}_i) - \lambda a = 0$$

$$\sum_{i=1}^n w_i (\mathbf{x}_i - \bar{\mathbf{x}})^\top \begin{pmatrix} b & c & d \\ c & -b & -a \\ d & a & -b \end{pmatrix} (\bar{\mathbf{m}} - \mathbf{m}_i) - \lambda b = 0$$

$$\sum_{i=1}^n w_i (\mathbf{x}_i - \bar{\mathbf{x}})^\top \begin{pmatrix} -c & b & a \\ b & c & d \\ -a & d & -c \end{pmatrix} (\bar{\mathbf{m}} - \mathbf{m}_i) - \lambda c = 0$$

$$\sum_{i=1}^n w_i (\mathbf{x}_i - \bar{\mathbf{x}})^\top \begin{pmatrix} -d & -a & b \\ a & -d & c \\ b & c & d \end{pmatrix} (\bar{\mathbf{m}} - \mathbf{m}_i) - \lambda d = 0$$

These can be combined in a single matrix equation

$$(\mathbf{A} - \lambda \mathbf{I}) \mathbf{a}^\top = 0$$

This equation is solved by choosing \mathbf{a} to be any eigenvector of \mathbf{A} . The solution that minimizes E'' is the eigenvector corresponding to the maximum eigenvalue

of \mathbf{A} (Horn, 1986). These quaternion values define the rotation matrix \mathbf{R} (Equation 2.10).

Thus both the translation and rotation have been determined giving the optimal pose that best maps the model to the measured 3D feature positions.

The number of templates tracked can be less than the total number. This allows the system to continue tracking when some templates suffer severe perspective distortion or are occluded altogether. The best templates to track can be determined from the estimated head pose as those that are visible and will appear most fronto-parallel to the image plane. Figure 2.27 shows the system in operation.

For our research we are interested in using existing head tracking technology to track the pose of the head, and then overlay the functionality to track deformable facial features. In Chapter 6 we use two of the head tracking systems described here. A monocular system based on Lowe's object tracking algorithm is used as the basis for a monocular lip-tracking system. Lowe's approach was chosen for this initial implementation owing to its simple and efficient implementation, robustness to noise in feature locations, and suitability for a monocular system. We then extend the work to a stereo system, and the stereo head tracker developed in our lab (Matsumoto and Zelinsky, 2000) and detailed above, is used as the basis for a stereo lip tracking system.

2.4.2 Tracking Deformable Facial Features

Tracking deformable facial features is a challenging problem, not only do the features move relative to the head, but they deform and change shape and appearance. The eyelids are an example of a deformable feature. We have already discussed detecting eyelid movement (blinks) in Section 2.1.4, however, as mentioned previously the movement of an eyelid is often too fast to be properly tracked by a 30Hz vision system. The mouth and eyebrows on the other hand are well suited for tracking by a 30Hz vision system. Tracking the mouth is the most challenging, as it displays a much wider range of deformation than the eyebrows, and exhibits drastic changes in appearance from open, closed, teeth visible, tongue visible, etc. states, as shown in Figure 2.28.

Mouth shape information is highly relevant to Human Computer Interaction, in particular verbal communication systems, and approaches applied to mouth



Figure 2.27: Example of Matsumoto and Zelinsky's system tracking the 3D pose of the head.

tracking are often transferable to tracking other deformable facial features, such as the eyebrows and eyelids. With these points in mind we have chosen to focus our study of deformable facial feature tracking on the problem of mouth tracking.

Verbal communication with computers offers a natural and intuitive alternative to keyboard and mouse interfaces. While these traditional interfaces offer precise and efficient means of inputting information, there are many circumstances where verbal interaction is preferable. Verbal interaction is hands-free, leaving the user's

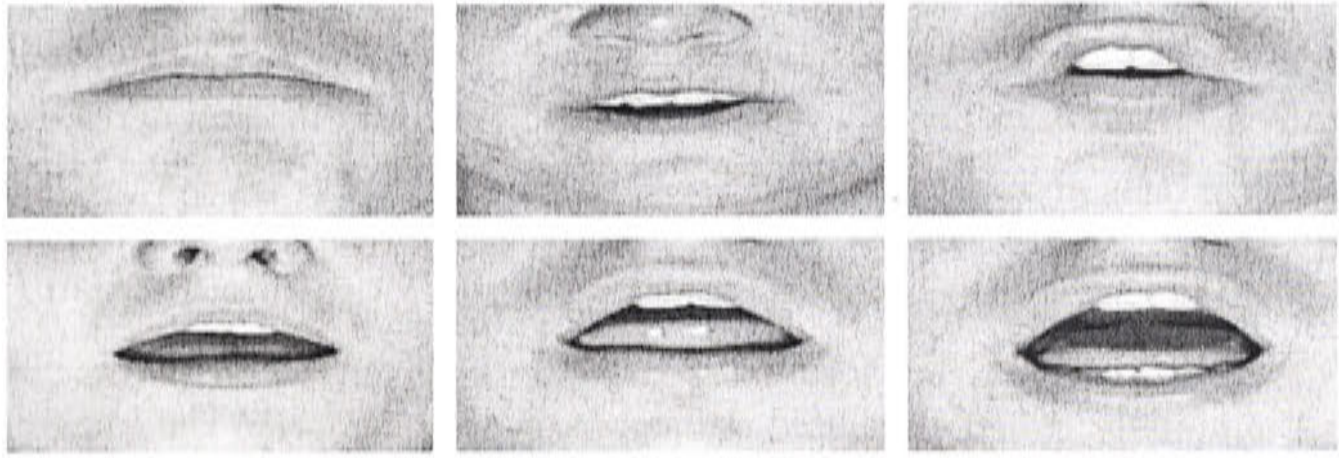


Figure 2.28: The appearance of a subject's mouth can vary greatly. Figure from Goecke (2002).

hands available for other tasks like driving a car or operating machinery. Verbal interaction can even remove the requirement for a keyboard altogether. Keyboards are bulky items, undesirable on small portable devices like pocket PCs or mobile phones. Disabled people unable to operate keyboards have found verbal communication invaluable for interacting with computers, and verbal communication has found use as a complement to keyboard and mouse interfaces, both for the ease that a user can input information, and the reduced risk of typing-related injuries such as RSI.

Whilst modern voice recognition systems have shown impressive recognition rates, and have led to successful commercial products, such as Dragon Naturally Speaking⁵, these systems require significant training, are user dependent, and do not perform well in the presence of noise. By using visual information to complement auditory input it is feasible to increase the robustness of automatic speech recognition systems.

Visual cues are important in speech. It is known that human perception of speech is enhanced when the face and mouth of the speaker are visible, and that the teeth, tongue and lips provide useful information regarding placement of articulation (Dodd and Campbell, 1987). To a certain extent even normal-hearing people lip-read as they listen to a speaker (McGurk and MacDonald, 1976) — although they may be unaware of the fact — and the shape of the mouth removes some of the ambiguity of the spoken words. Realtime computer lip-reading can provide this same visual information for automatic speech recognition systems.

Combining auditory and visual cues for speech processing, known as audiovisual

⁵<http://www.dragonsys.com/naturallyspeaking/>

speech processing, has been a fertile field of research over the last decade, and numerous systems have been developed; see Chen (2001) for a general overview of research in this area.

The first audiovisual speech reading system was put forward by Petajan (1985). This system used a single monochrome camera and custom video processing hardware. A manually tuned threshold value was used to binarize the image so the nostrils and mouth could be identified, and parameters such as mouth area, height and width were determined. The system was tested on isolated utterances and it was demonstrated that visual information obtained was beneficial for recognition. Since then many more approaches have been adopted for tracking lips, these range from simple image-based methods like integral projection Yang *et al.* (1998a), to active contours Kaucic *et al.* (1996) and complex 3D lip models Revéret and Benoît (1998). Some methods have required the subject to wear special coloured lipstick (Kaucic *et al.*, 1996; Adjoudani and Benoît, 1996; Benoît *et al.*, 1996), whilst others are able to track the unadorned lip contour (Revéret and Benoît, 1998).

In 1996 Kaucic *et al.* presented an automatic lip-reading system that enhanced the performance of speech recognition on a forty word vocabulary beyond that achieved using a purely audio-based approach. Whilst the addition of visual information was only marginally beneficial in the noise free case, in the presence of noise (with a signal to noise ratio of -3dB) the error rate was reduced significantly. This system ran in realtime. It tracked the mouth from a frontal view, but required the user to wear lipstick to enhance the contrast between the lips and the surrounding skin.

Kaucic *et al.* also considered an alternative lip tracker that tracked the silhouette of the lips from a profile view of the subject, and was capable of realtime tracking without cosmetic aids. This profile tracker benefited from the strong contrast the lip silhouette made with the white background placed behind the subject, however, the teeth and tongue were not visible in the profile view, nor was it possible to determine the mouth width, and owing to the importance of these features in visual lip reading the profile approach was dropped in favour of the frontal-view tracker.

Both Kaucic *et al.*'s front-on and profile trackers used active contours to track the lip contour. Quadratic *B*-splines were used to model the contour, and the dynamics of the models were learned using a Maximum Likelihood Estimation

algorithm. A Kalman filter was used to blend the predicted and observed lip locations. Experiments were performed separately with the profile tracker and the frontal view tracker (with lipstick) to investigate the extent to which lip contour information improved speech recognition. However, no experiments were reported using both profile *and* frontal view information together, so all visual information is based on lip contour locations in a single 2D plane.

The LAFTER (Lips and Face Real Time Tracker) from MIT Media Lab (Oliver *et al.*, 1997) uses an entirely different approach to lip tracking. Here blobs rather than contours are used to represent features. Blob segmentation offers an alternative to using edge features where the regions themselves, not the edges, form the features. A blob is defined as a compact set of pixels that share a visual property, such as colour, texture, motion, or a combination thereof, which differentiates them from the surrounding image. These blobs are represented by their low order statistics, that is, the mean and covariance of the pixels in the blob, and mixture of Gaussian distributions are used to model the blob features in colour space.

The initial mixture parameters for each blob feature are determined using an off-line training process on large training sets of faces, lips and mouth cavities. At start up the face and mouth features are located using these general mixture models. These models are later adapted on-line during tracking. A Gaussian mixture model is also constructed for the background, and this is learnt entirely on-line.

The face and mouth are found using colour and shape information. The normalised *rg* colour components of pixels (see Section 2.1.2) are examined to determine those that could potentially belong to each feature, and blob models are chosen that best describe the shape of the dominant pixel clusters. The system uses an active camera to ensure the user always appears at the desired size in the centre of the image. Both the face and mouth are modelled using Gaussian mixture models describing the chromatic colour and spatial distribution of their constituent pixels. Thus the mouth is characterised by its area, spatial eigenvalues and bounding box. The resulting mouth characteristics are made invariant to rotation about the optical axis by determining the orientation of the face in the image plane, however, the result is not robust to rotation of the subject's head out of the image plane.

Revéret and Benoît (1998) present a method for modeling lip shape as a 3D

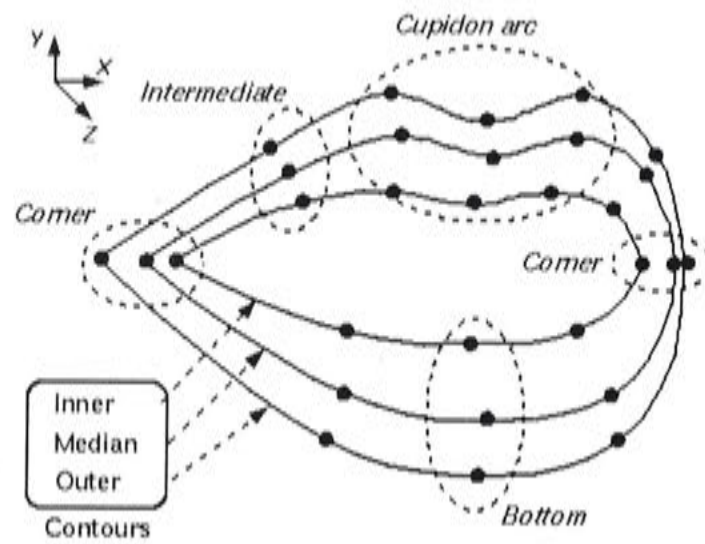


Figure 2.29: Revéret and Benoît's lip model showing the 3 polynomial contours each defined by 10 control points (Revéret and Benoît, 1998).

polynomial surface. This method is applied to lip tracking, modelling lip motion in speech production, and for visual speech animation of lip motion during speech. The model is shown in Figure 2.29 and consists of 3 polynomial contours each defined by 10 control points. The system is trained for each subject, using a graphical user interface to fit the lip model onto calibrated front and profile images of the mouth, as the subject forms ten key lip shapes (chosen based on phonetic observations of spoken French). The 90 (x, y, z) locations of the lip model control points observed for each of the key lip shapes are treated as feature vectors, principal component analysis is applied across the ten key lip shapes, and three principal lip shape feature vectors are extracted. It is shown that linear combinations of these three principal components account for 94% of the lip shape variability exhibited across the ten key lip shapes used for training, so by restricting the model to only forming lip shapes that are such linear combinations, the model may be described by just three parameters.

This three-parameter model is used in conjunction with colour segmentation to identify the 3D lip shape in a monocular image sequence. Colour segmentation is done using a lip colour model that is built from the training data, and normalised rg chrominance is used to afford some robustness to lighting variations. The system was tested on a single phonetically balanced sentence and the results show good recovery of internal and external lip contours. The main drawback of this system was the significant training required for each new subject. Also, while the modelling approach allows the 3D shape of the lip to be estimated from the monocular front-on images used for tracking, the lip shape was restricted by the range of motion displayed in the training data, and during tracking the 3D

shape is being *modelled* rather than *measured*.

The numerous systems available offer varying solutions to the challenging problem of lip-tracking. However, little interest has been shown by researchers in recovering raw 3D information about the mouth shape. While it is debateable how beneficial such information is for speech processing, it is certainly important for character animation and tracking the true 3D shape of the mouth. Another important factor for a practical lip-tracking system is the ability to deal with head motion, in particular the apparent deformation of the mouth caused by the subject turning his or her head while speaking. While the LAFTER system can accommodate some head movement neither LAFTER nor Kaucic *et al.*'s system corrects for distortion of mouth shape due to the head rotating out of the image plane.

In Chapter 6 we present two lip tracking systems, one monocular and one stereo, both operating on grey-scale images, and running in conjunction with a head tracker to enable the system to perform robustly through a range of head poses. The monocular system measures lip height and width, while the stereo system recovers the raw 3D locations of points on the lip contour. The stereo lip tracker is the first system to use stereo to directly measure the 3D locations of tracking points on the mouth and track the raw 3D mouth shape.

2.5 Summary

The human face has a distinctive and unique appearance. There are a number of qualities governing the appearance of a face that distinguish it from other objects, and several of these can be automatically detected by computer vision systems. Face regions can be identified by their size, shape and colour together with the occurrence of facial features that appear as dark blobs or peaks of radial symmetry at particular locations. Motion detection, depth estimation, colour segmentation and radial symmetry detection all provide visual cues for detecting different facial qualities. Radial symmetry is a particularly attractive cue, however the best results from existing transforms are slow and inefficient to compute. In Chapter 3 we present a new, computationally efficient method for determining radial symmetry whose results rival or surpass those of existing methods whilst being fast enough to operate in realtime.

Enabling a computer to see a face can be considered as a three step process, face localisation to detect approximately where the face is in a scene, face registration to register the facial features, and face tracking to track the pose of the head and the movement of deformable facial features.

Particle filtering and the use of multiple cues have been shown to be very effective for robust face localisation. Particle filtering enables the tracking of multiple hypotheses and provides an efficient means of searching a multi-dimensional state space. Multiple cues enable a system several modalities with which to detect a target, increasing robustness to changes in tracking conditions and allowing the system scope to adapt to such changes. In a realtime system it is important to consider the allocation of computational resources across different cues. In Chapter 4 we present a realtime face localisation system that uses particle filtering, multiple cues, and efficiently allocates computational resources across cues according to the quality of information being produced by each cue.

Integral projection provides an efficient way of detecting facial features for face registration in a sufficiently constrained situation, while other methods such as blink detection, and filtering techniques for enhancing features are equipped to deal with a wider range of inputs scenarios. Chapter 5 presents a face registration system capable of verifying the presence of a face and detecting facial features. The system uses blink detection to initially localise the eye and face location, then filtering and radial symmetry detection to enhance facial features, and finally feature locations will be pin-pointed using integral projection.

Face tracking involves tracking both rigid and deformable facial features, in order to describe both the 3D pose of the head and the shape and location of facial features. Both monocular and stereo systems are available for tracking the pose of the head, and tracking the mouth (the dominant deformable feature) has received considerable attention over the last decade. However, virtually all systems track using a single camera, and while some infer the 3D shape of the mouth from learnt models they do not track the mouth in 3D as is possible with a stereo system. In Chapter 6 we present both monocular and stereo lip tracking systems, both running in conjunction with head trackers to facilitate robust performance during head motion. Our stereo system directly measures the mouth shape in 3D during tracking, enabling unconstrained 3D tracking of this deformable feature.

Chapter 3

Fast Radial Symmetry Detection

IN the previous chapter we explained how useful radial symmetry can be for detecting facial features in images (Section 2.1.5), and we reviewed an extensive list of existing methods for calculating radial symmetry in images. The best results for facial feature detection came from the generalized symmetry transform (Reisfeld and Yeshurun, 1998), Figure 2.13. However, this transform is slow, computationally expensive to compute, and not well-suited to realtime applications. Other methods provide more efficient alternative techniques for computing radial symmetry, but the results obtained are not as useful for locating facial features. In this chapter we will develop a new, computationally efficient method for determining radial symmetry that is able to produce results that rival those from the generalized symmetry transform, and other existing methods, whilst being fast enough to operate in realtime.

We present a novel gradient-based interest operator, the Fast Radial Symmetry Transform (FRST), that efficiently detects points of high radial symmetry. Our initial approach was inspired by the results of the generalized symmetry transform (Reisfeld *et al.*, 1995; Intrator *et al.*, 1995; Reisfeld and Yeshurun, 1998), although the final method bears more similarity to the work of Sela and Levine (1997) and the circular Hough transform (Kimme *et al.*, 1975; Minor and Sklansky, 1981). The FRST determines the contribution each pixel makes to the symmetry of pixels around it, rather than considering the contribution of a local neighbourhood to a central pixel. Unlike previous techniques that have used this approach (Kimme *et al.*, 1975; Minor and Sklansky, 1981; Sela and Levine, 1997) it does not require the gradient to be quantized into angular bins, the contribution of *every* orientation is computed in a single pass over the image. The FRST works

well with a general fixed parameter set, however, it can also be tuned to exclusively detect particular kinds of features. Computationally the algorithm is very efficient, being of order $O(KN)$ when considering local radial symmetry in $N \times N$ neighbourhoods across an image of K pixels.

In Section 3.1 of this chapter we define the FRST. Section 3.2 discusses the selection of parameters, and Section 3.3 describes how the transform can be adapted to different tasks. Section 3.4 presents several general sets of parameters suitable for different applications of the FRST. Section 3.5 shows the performance of the FRST on a variety of images, and compares it to existing techniques, and Section 3.6 presents the conclusions.

3.1 Definition of the Transform

The FRST is calculated at one or more radii $n \in N$, where N is the set of radii of the radially symmetric features to be detected. The value of the transform at radius n indicates the contribution to radial symmetry of the gradients a distance n away from each point. Whilst the transform can be calculated for a continuous set of radii this is generally unnecessary as a subset of radii is normally sufficient to yield a representative result.

The algorithm can be summarised as follows, and is discussed in detail hereafter:

1. Determine gradient \mathbf{g} .
2. For each radius under consideration:
 - (a) Consider each gradient element in turn, for each gradient element:
 - i. Determine affected pixels.
 - ii. Calculate orientation and magnitude projection images \mathbf{O}_n and \mathbf{M}_n .
 - (b) Combine \mathbf{O}_n and \mathbf{M}_n to form the unfiltered symmetry contribution \mathbf{F}_n
 - (c) Calculate the symmetry image at radius n , \mathbf{S}_n by blurring \mathbf{F}_n via convolution with \mathbf{A}_n .
3. Sum \mathbf{S}_n over all radii $n \in N$ to determine the final symmetry image \mathbf{S} .

An overview of the algorithm is shown in Figure 3.1 showing the key steps to generating the output radial symmetry image \mathbf{S} from the grey-scale input image \mathbf{I} . The figure also shows some example images illustrating the output of different stages of the process. The remainder of this section will work through the different stages of the algorithm.

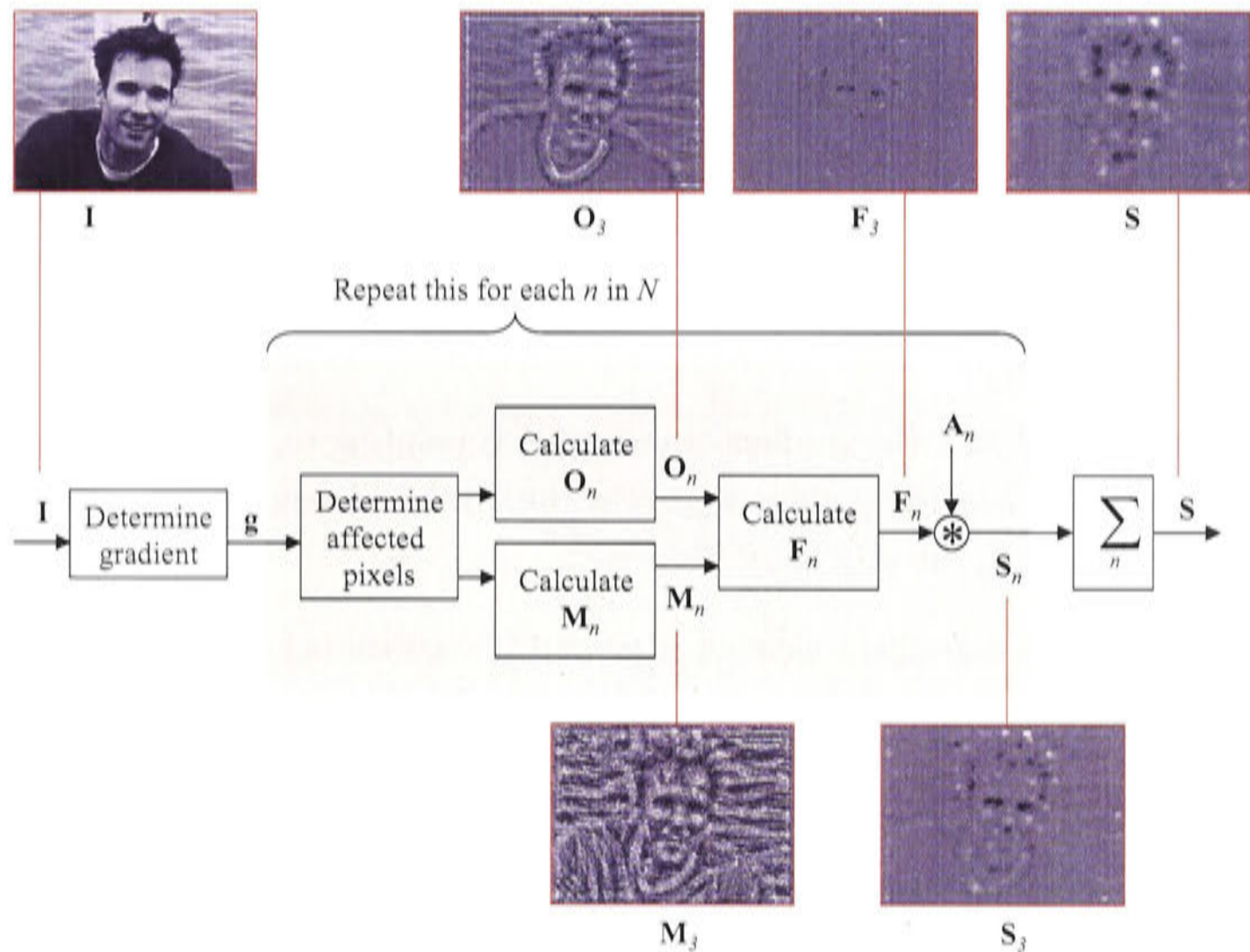


Figure 3.1: Steps involved in computing the FRST. Example images are also shown throughout the process, positive values are shown as light pixels, negatives as dark and zero as mid-grey, gradient is assumed to point from dark to light.

Initially the gradient of the image \mathbf{I} is determined, for our experiments we used a 3×3 Sobel operator

$$\mathbf{K} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

to determine the gradient in the x and y directions as $\mathbf{I} * \mathbf{K}$ and $\mathbf{I} * \mathbf{K}^\top$ respectively.

For each radius n that is being considered we determine the *affected pixels* by examining the gradient. Each non-zero gradient element generates a *positively affected pixel* and a *negatively affected pixel*. The *positively-affected pixel* is de-

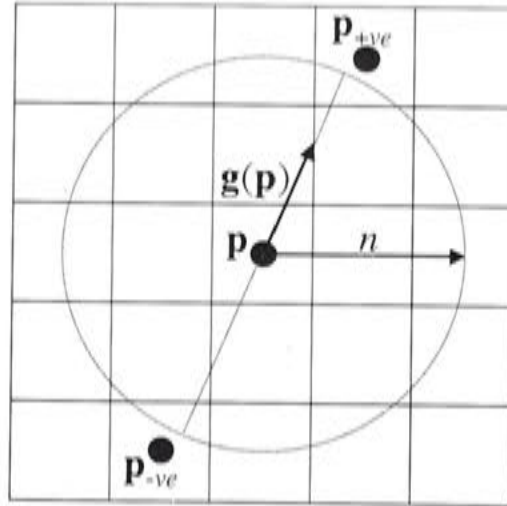


Figure 3.2: The locations of pixels $\mathbf{p}_{+ve}(\mathbf{p})$ and $\mathbf{p}_{-ve}(\mathbf{p})$ affected by the gradient element $\mathbf{g}(\mathbf{p})$ for a range of $n = 2$. The circle shows all the pixels which can be affected by the gradient at \mathbf{p} for a radius n .

defined as the pixel that the gradient vector $\mathbf{g}(\mathbf{p})$ is pointing to, a distance n away from \mathbf{p} , and the *negatively-affected pixel* is the pixel a distance n away that the gradient is pointing directly away from.

Figure 3.2 shows a gradient element $\mathbf{g}(\mathbf{p})$ and its associated positive and negatively affected pixels labelled \mathbf{p}_{+ve} and \mathbf{p}_{-ve} respectively.

Formally the coordinates of the positively-affected pixel are given by

$$\mathbf{p}_{+ve}(\mathbf{p}) = \mathbf{p} + \text{round} \left(\frac{\mathbf{g}(\mathbf{p})}{\|\mathbf{g}(\mathbf{p})\|} n \right)$$

while those of the negatively-affected pixel are

$$\mathbf{p}_{-ve}(\mathbf{p}) = \mathbf{p} - \text{round} \left(\frac{\mathbf{g}(\mathbf{p})}{\|\mathbf{g}(\mathbf{p})\|} n \right)$$

where “round” rounds each vector element to the nearest integer.

Next we use these affected pixels to form an *orientation projection image* \mathbf{O}_n and a *magnitude projection image* \mathbf{M}_n . Initially the orientation and magnitude projection images are zero. For each pair of affected pixels the corresponding point \mathbf{p}_{+ve} in the orientation projection image \mathbf{O}_n and magnitude projection image \mathbf{M}_n is incremented by 1 and $\|\mathbf{g}(\mathbf{p})\|$ respectively, while the point corresponding to \mathbf{p}_{-ve} is decremented by these same quantities in each image. That is

$$\mathbf{O}_n(\mathbf{p}_{+ve}(\mathbf{p})) = \mathbf{O}_n(\mathbf{p}_{+ve}(\mathbf{p})) + 1$$

$$\mathbf{O}_n(\mathbf{p}_{-ve}(\mathbf{p})) = \mathbf{O}_n(\mathbf{p}_{-ve}(\mathbf{p})) - 1$$

$$\mathbf{M}_n(\mathbf{p}_{+ve}(\mathbf{p})) = \mathbf{M}_n(\mathbf{p}_{+ve}(\mathbf{p})) + \|\mathbf{g}(\mathbf{p})\|$$

$$\mathbf{M}_n(\mathbf{p}_{-ve}(\mathbf{p})) = \mathbf{M}_n(\mathbf{p}_{-ve}(\mathbf{p})) - \|\mathbf{g}(\mathbf{p})\|$$

Once \mathbf{O}_n and \mathbf{M}_n have been calculated the radial symmetry contribution at radius n can be determined from the convolution

$$\mathbf{S}_n = \mathbf{F}_n * \mathbf{A}_n \quad (3.1)$$

where \mathbf{F}_n is the unfiltered symmetry contribution defined by

$$\mathbf{F}_n(\mathbf{p}) = \frac{\mathbf{M}_n(\mathbf{p})}{k_n} \left(\frac{|\tilde{\mathbf{O}}_n(\mathbf{p})|}{k_n} \right)^\alpha \quad (3.2)$$

and

$$\tilde{\mathbf{O}}_n(\mathbf{p}) = \begin{cases} \mathbf{O}_n(\mathbf{p}) & \text{if } \mathbf{O}_n(\mathbf{p}) < k_n \\ k_n & \text{otherwise} \end{cases} \quad (3.3)$$

\mathbf{A}_n is a two-dimensional Gaussian, α is the radial strictness parameter, and k_n is a scaling factor that normalizes \mathbf{M}_n and \mathbf{O}_n across different radii. These parameters are discussed in more detail in Section 3.2.

The full transform is defined as the average of the radial symmetry contributions over all the radii considered,

$$\mathbf{S} = \frac{1}{|N|} \sum_{n \in N} \mathbf{S}_n \quad (3.4)$$

If the gradient is calculated to point from dark to light then the output image \mathbf{S} will have positive values corresponding to bright radially symmetric regions and negative values indicating dark symmetric regions as can be seen in Figure 3.1.

It can be more useful to consider the gradient orientation exclusively, removing the effect of contrast on the level of interest attributed to points in the image.

This leads to an alternate *orientation-based* radial symmetry that is defined by replacing \mathbf{F}_n in Equation 3.1 by

$$\hat{\mathbf{F}}_n(\mathbf{p}) = \text{sgn}(\tilde{\mathbf{O}}_n(\mathbf{p})) \left(\frac{|\tilde{\mathbf{O}}_n(\mathbf{p})|}{k_n} \right)^\alpha$$

This provides a result that is more robust to lighting changes. However, when applying this orientation-based formulation it is generally necessary to ignore small gradients that tend to add noise to the result, this is discussed in detail in Section 3.3.1.

3.2 Choosing the Parameters

The definition of the transform contains a number of parameters which need to be appropriately defined, these are:

- a set of radii $N = \{n_1, n_2, \dots\}$ at which to calculate \mathbf{S}_n ,
- the Gaussian kernels \mathbf{A}_n ,
- the radial strictness parameter α , and
- the normalizing factor k_n .

This section discusses each of these in turn and describes their effect on the output of the transform. A general set of parameters is presented in Section 3.4.

3.2.1 Set of Radii N

The traditional approach to local symmetry detection (Di Gesù and Valenti, 1995a; Reisfeld *et al.*, 1995; Sela and Levine, 1997) is to calculate the symmetry apparent in a local neighbourhood about each point. This can be achieved by calculating S_n for a continuous set of radii $N = \{1, 2, \dots, n_{max}\}$ and combining using Equation 3.4. However, since the symmetry contribution is calculated independently for each radius n it is simple to determine the effects at a single radius, or an arbitrary selection of radii that need not be continuous. Furthermore, the results obtained by only examining alternate radii give a good approximation to the output obtained by examining all the radii, while saving on computation.

The effect of choosing sparse sets of radii was quantified experimentally by comparing the output of the transform calculated across all radii from 1 to 5 to that calculated across several sparse sets of radii. Labelling the output image calculated across radii 1 to 5 as $\mathbf{S}_{1...5}$ and the sparse outputs calculated across radii N as \mathbf{S}_N the error is defined as

$$\mathbf{E}_N(\mathbf{p}) = \mathbf{S}_N(\mathbf{p}) - \mathbf{S}_{1...5}(\mathbf{p})$$

The experiment was run over a database of 295 diverse face images, and the average power of the error \mathbf{E}_N was determined for each set of radii. The results are shown in Table 3.1 with the power of the error expressed as a percentage of the power of the transform calculated across all five radii ($\mathbf{S}_{1...5}$).

Table 3.1: Parameter Settings used for Experimentation

Radii	1,2,3,4,5	1,2,3	1,5	3
Error Power	0%	7.9%	37%	780%

Table 3.1 shows that taking alternative radii (1, 3, 5) gives a close approximation to using all the radii with an error of only 7.9% between the two outputs. Unsurprisingly, as less radii are included the error increases quite rapidly.

An example of this experiment is shown in Figure 3.3. The small error between the transform calculated at alternative radii $\mathbf{S}_{1,3,5}$ and $\mathbf{S}_{1...5}$ is evident with $\mathbf{E}_{1,3,5}$ being close to zero. Exactly how close an approximation is achieved by using only alternate radii depends on the edges of the features being detected. Sharp-edged features are more likely to be attenuated if the transform is not calculated at their exact radius.

If the scale of a radially symmetric feature is known *a priori* then the feature can be efficiently detected by only determining the transform at the appropriate radii. For example, the irises in the eyes in the input image in Figure 3.3 have a radius of approximately 5 pixels, so will be well detected using a radius of 5, or radii 1 and 5 as can be seen in Figure 3.3.

3.2.2 Gaussian Kernels \mathbf{A}_n

The purpose of the Gaussian kernel \mathbf{A}_n is to spread the influence of the positively- and negatively-affected pixels as a function of the radius n . A rotation invariant

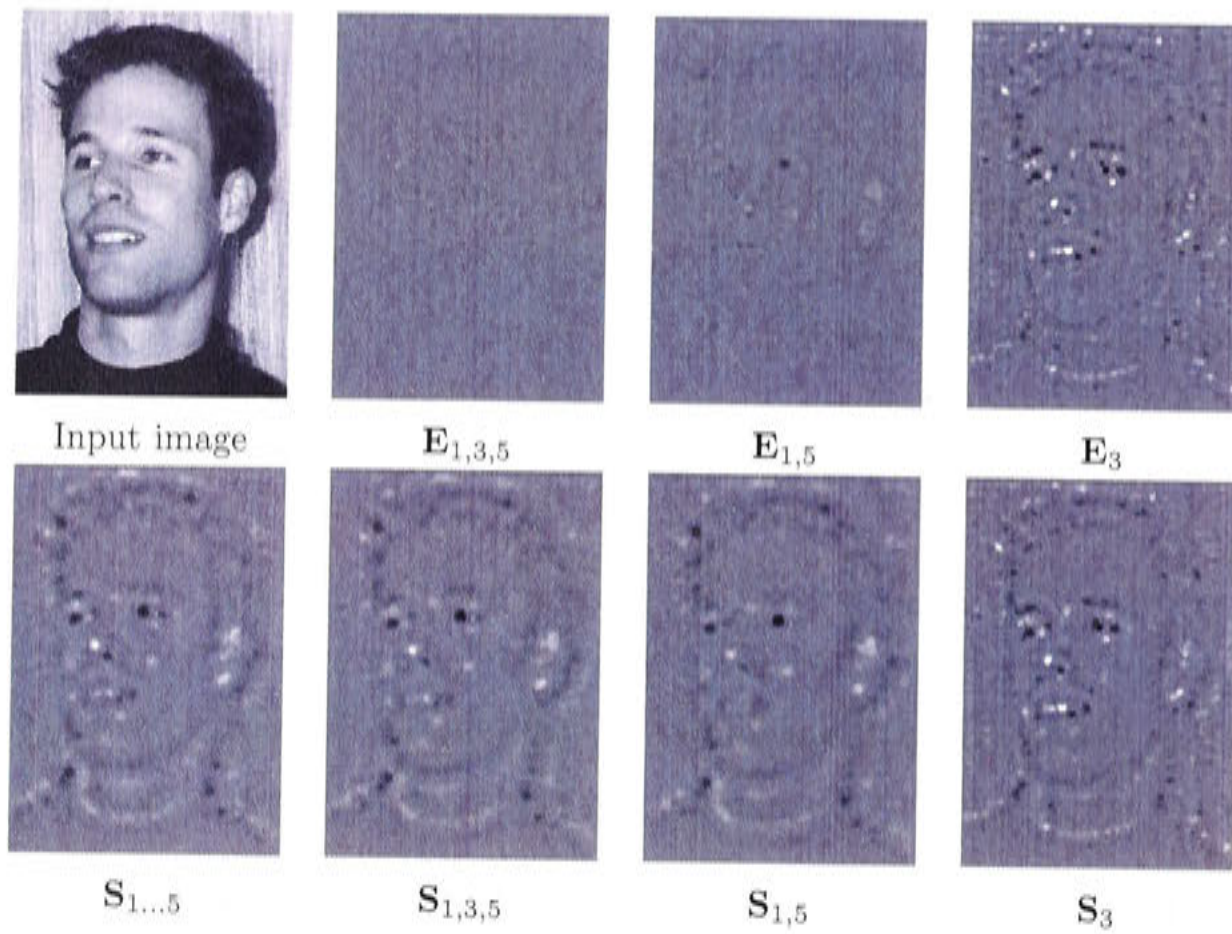


Figure 3.3: Effect of varying the set of radii N at which the FRST is computed. S_N is the output of the transform and E_N is the error, positive values are shown as light pixels, negative as dark, and zero as mid-grey.

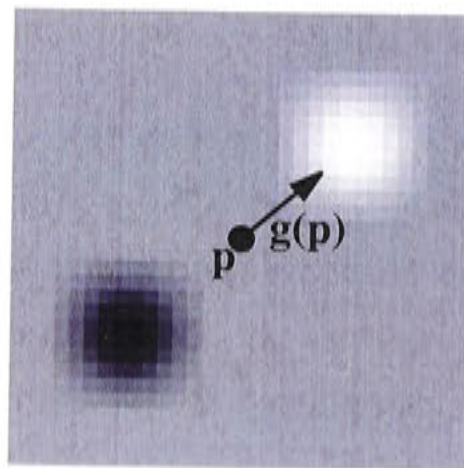


Figure 3.4: The contribution of a single gradient element, with A_n chosen to be a 2D Gaussian of size $n \times n$ and standard deviation $\sigma = 0.25n$, and $n = 10$.

two-dimensional Gaussian is chosen since it has a consistent effect over all gradient orientations, and it is separable so its convolution can be efficiently determined. Figure 3.4 shows the contribution for a single gradient element $\mathbf{g}(\mathbf{p})$. By scaling the standard deviation linearly with the radius n , an arc of influence is defined that applies to all affected pixels. The width of the arc is defined by scaling the standard deviation of \mathbf{A}_n with respect to n .

All \mathbf{A}_n are defined as two dimensional Gaussians whose elements sum to n . Convolution with \mathbf{A}_n has the result of spreading the effect of each gradient element by an amount proportionate to the standard deviation of the Gaussian, and amplifying its magnitude by n . Amplifying the magnitude is necessary to prevent the effect of gradient elements becoming negligible at large radii as a result of being spread too thinly across the image.

Even though the convolution with the Gaussian kernel is separable, depending on the size of the kernel used, this is often still the most time consuming part of the algorithm. This step can be sped up by replacing the Gaussian kernels with uniformly flat kernels¹ whose convolution can be calculated recursively. Surprisingly this still yields reasonable results in most circumstances, however, uniform (square) kernels are not invariant to rotation so Gaussian kernels are preferred. All results, with the exception of the realtime results in Figure 3.13, in this thesis are obtained using Gaussian kernels.

3.2.3 Radial-strictness Parameter α

The parameter α determines how strictly radial the radial symmetry must be for the transform to return a high interest value. Figure 3.5(a) illustrates the effect of α on \mathbf{O}_n at the pixel level. Note how a higher α strongly attenuates the line relative to the dot.

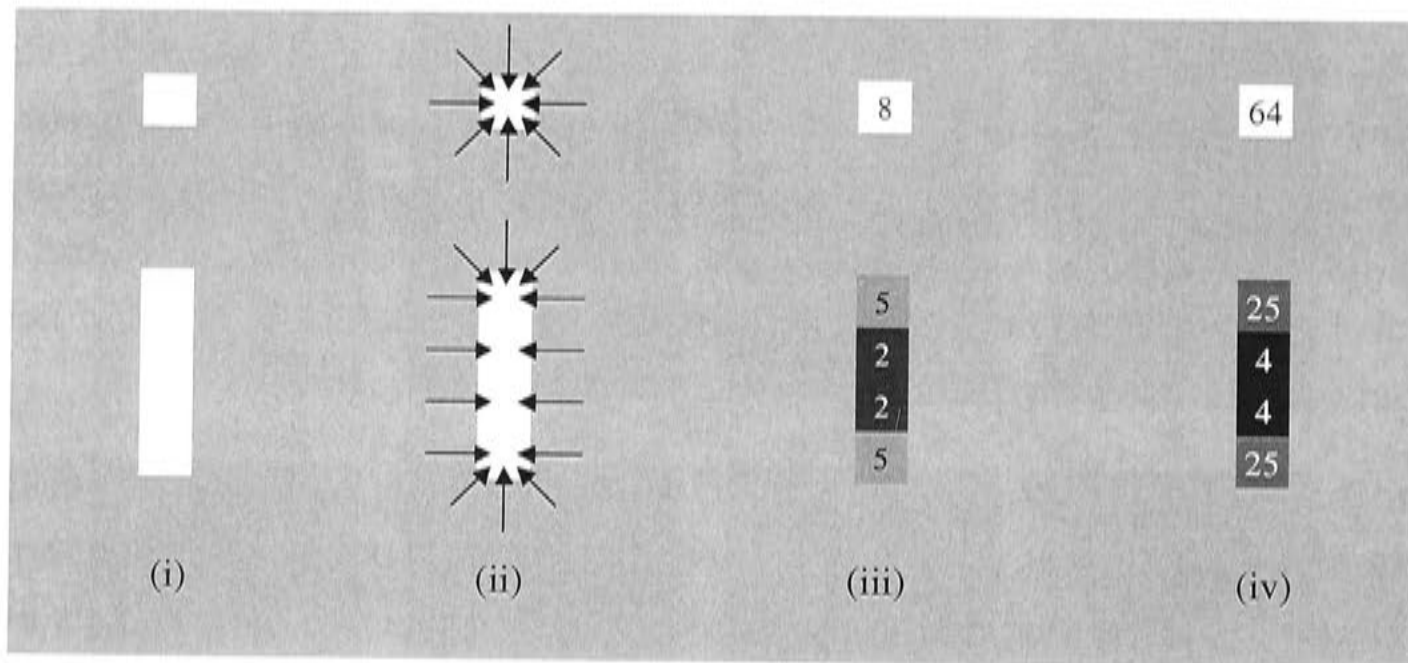
Figure 3.5(b) shows the effect of choosing α to be 1, 2 and 3 on S_1 for an image exhibiting strong radial values around the eyes. Once again a higher α eliminates non-radially symmetric features such as lines.

A choice of $\alpha = 2$ is suitable for most applications. Choosing a higher α starts attenuating points of interest, whilst a lower α gives too much emphasis to non-radially symmetric features, however, choosing $\alpha = 1$ minimizes the computation when determining \mathbf{F}_n in Equation 3.2.

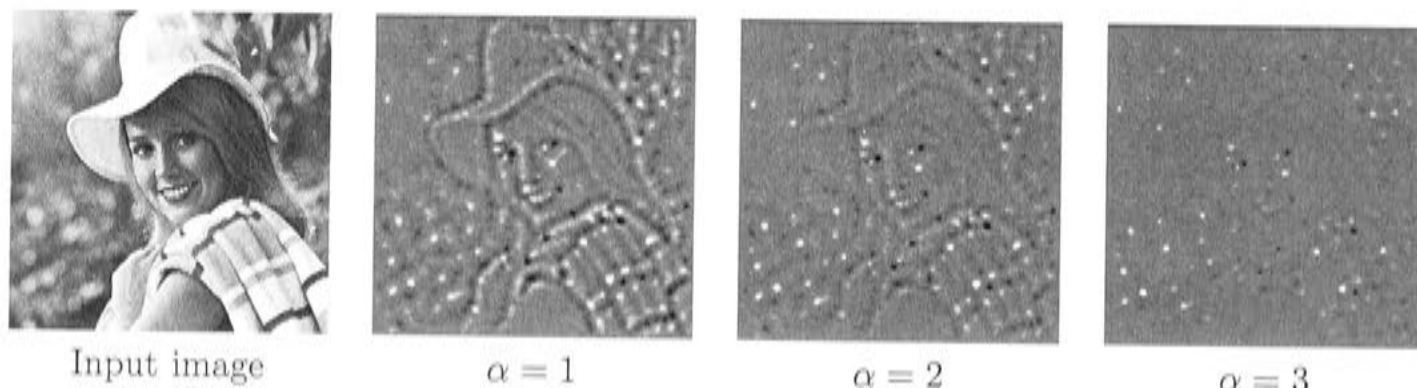
3.2.4 Normalizing Factor k_n

In order to compare or combine the symmetry images calculated for different radii they must be represented on a similar scale. As the radius increases so does the

¹A *uniformly flat kernel* refers to a convolution kernel (matrix) whose elements are all the same value.



(a)



(b)

Figure 3.5: Effect of varying α (a) At the pixel level (i) Sample arrangement of light pixels on a dark background, (ii) Gradient from adjacent pixels, (iii) Number of gradient elements pointing at each pixel O_n , (iv) Square of the number of gradient elements pointing at each pixel O_n^2 . (b) Effect of varying α at the image level. Original image from the USC-SIPI Image Database.

number of gradient elements that could potentially effect each pixel, that is, the number of pixels on the perimeter of the circle in Figure 3.2.

One way of normalizing across scales is to divide O_n and M_n through by their maximum values. However, this scales the result at each radius relative to itself, and does not provide an absolute measure that can be used to compare between different radii or different images.

It is preferable to scale O_n and M_n by the *expected* maximum value of O_n , and saturate O_n at this value, as is done in Equation 3.3, restated here,



Figure 3.6: Some example images from the test set.

$$\tilde{\mathbf{O}}_n(\mathbf{p}) = \begin{cases} \mathbf{O}_n(\mathbf{p}) & \text{if } \mathbf{O}_n(\mathbf{p}) < k_n \\ k_n & \text{otherwise} \end{cases}$$

\mathbf{M}_n cannot be saturated in the same way, (although it could be averaged using division by \mathbf{O}_n) however, large values of \mathbf{M}_n do not cause problems, since the elements of \mathbf{O}_n are raised to an exponential power and so become much more significant than \mathbf{M}_n at locations where \mathbf{O}_n saturates.

Determining the expected maximum value of \mathbf{O}_n is best done experimentally, since it depends on gradient directions of neighbouring pixels and these gradient elements are not probabilistically independent. An experiment was conducted to determine the mean maximum value of \mathbf{O}_n for a set of 295 real images for $n = 1$ to 30. The set of test images comprised of photographs of people at a range of scales, and with widely varying backgrounds and lighting conditions. All images were in JPEG format and were obtained off the internet, image size varied from 108×130 to 405×244 pixels. The 295 images used for this experiment are contained on the CD-ROM enclosed with this thesis, and some examples are shown in Figure 3.6.

The orientation projection images \mathbf{O}_n were determined for each image (as described in Section 3.1) at all radii $n \in \{1, \dots, 30\}$. The maximum value of each \mathbf{O}_n was then determined giving a set of 30 maximum values m_n . This was re-

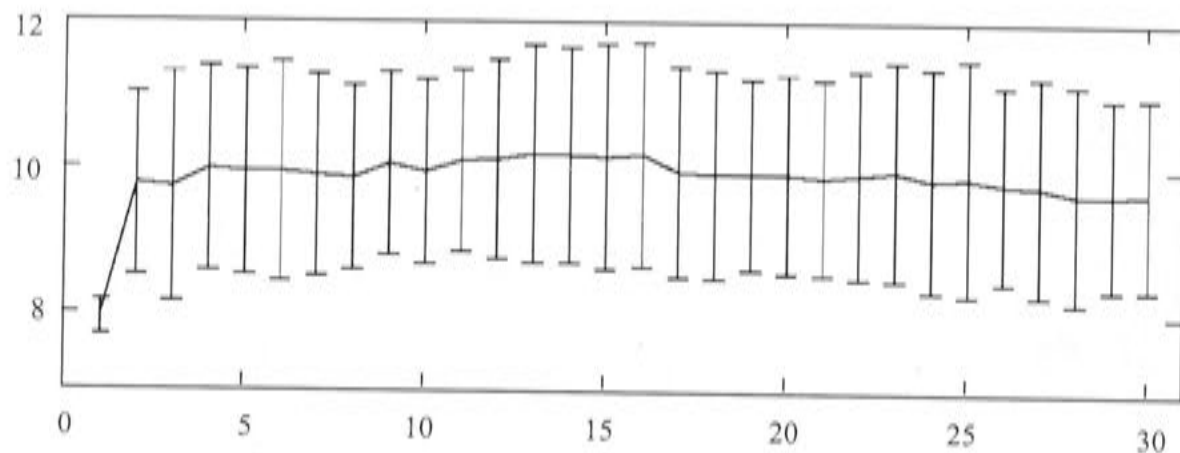


Figure 3.7: The mean and standard deviation of the maximum value of the orientation projection images \mathbf{O}_n for $n = 1$ to 30, calculated over 295 images.

peated for every image in the test set and the resulting sets of maximum values combined to determine the mean (expected) maximum value of \mathbf{O}_n for a given $n \in \{1, \dots, 30\}$.

The result is shown in Figure 3.7. Apart from the value of 8 for $n = 1$ (there are only 8 pixels a distance 1 away from any pixel) the expected values for all radii $n \in [2, 30]$ lay within $9.9 \pm 3\%$.

Using

$$k_n = \begin{cases} 8 & \text{if } n = 1 \\ 9.9 & \text{otherwise} \end{cases}$$

suitably normalizes \mathbf{M}_n and \mathbf{O}_n in Equation 3.2.

3.3 Refining the Transform

The transform can be refined to further increase computational speed and detect particular kinds of features. Refinements include:

- ignoring small gradients when calculating \mathbf{O}_n and \mathbf{M}_n .
- calculating dark (bright) symmetry by ignoring negatively- (positively-) affected pixels when determining \mathbf{O}_n and \mathbf{M}_n .
- Choosing a constant \mathbf{A}_n .

3.3.1 Ignoring Small Gradients

Gradient elements with small magnitudes have less reliable orientations, are more easily corrupted by noise, and tend to correspond to features that are not immediately apparent to the human eye. Since the purpose of the transform is to pick out points of interest in the image it is logical to ignore such elements in our calculation. Reisfeld implemented the generalised symmetry transform to ignore small gradients in his original work (Reisfeld, 1993), and Sela and Levine (1997) also ignore small gradient elements. They also go one step further and binarize the gradient image into an edge map.

We ignore small gradients by introducing a gradient threshold parameter β . When calculating images \mathbf{O}_n and \mathbf{M}_n all gradient elements whose magnitudes are below β are ignored. The effect of a small β on \mathbf{M}_n is negligible, however, even small values of β start to attenuate \mathbf{O}_n in regions of low contrast. This results in an emphasis on interest points with high contrast.

A small value of β that eliminates the lowest 1 – 2% of the gradient removes the small noisy gradients mentioned above. However, if low contrast features are not important, larger values of β can be chosen to increase the speed of the algorithm by considering fewer gradient elements. The effect of large values of β is shown in Figure 3.8, where β is measured as a percentage of the maximum possible gradient magnitude. In this example this is beneficial for the detection of eyes and mouth with $\beta = 20\%$, however, too high a value, such as $\beta = 40\%$ starts to attenuate features of interest such as the corners of the mouth. In general a conservative choice of $\beta = 2\%$ is preferable, higher values should only be used when it is desired to ignore low contrast features.

3.3.2 Dark & Bright Symmetry

The transform can be tuned to detect dark or bright regions of symmetry. To find dark regions exclusively only the negatively-affected pixels need be considered when determining \mathbf{M}_n and \mathbf{O}_n . Likewise, to detect bright symmetry only positively-affected pixels need be considered.

Alternatively, dark and bright symmetries can be obtained applying a threshold to the output image \mathbf{S} to eliminate all positive or negative values. This second approach has the advantage that inconsistent dark and light values will can-

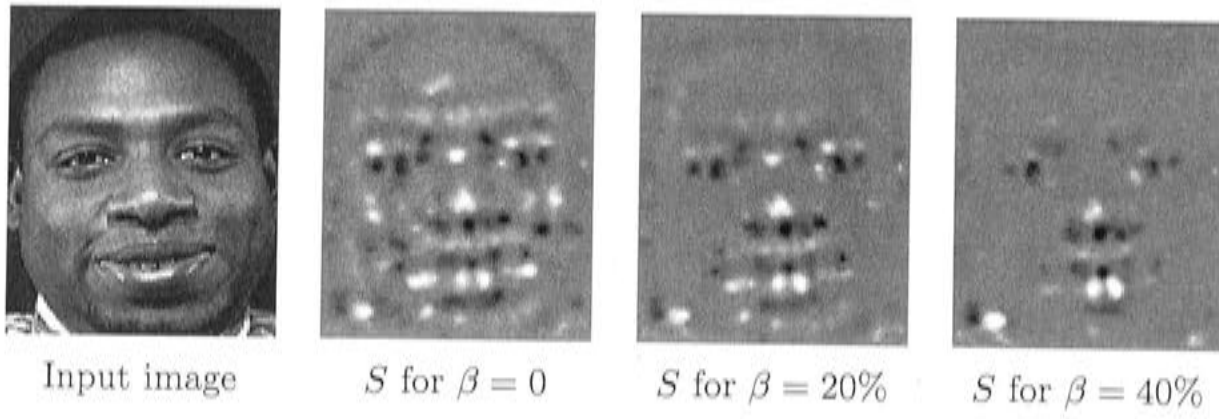


Figure 3.8: The effect of different values of β on S . Here β is measured as a percentage of the maximum possible gradient magnitude and $n = 1$. Original image from Database of Faces, AT&T Laboratories Cambridge 1994

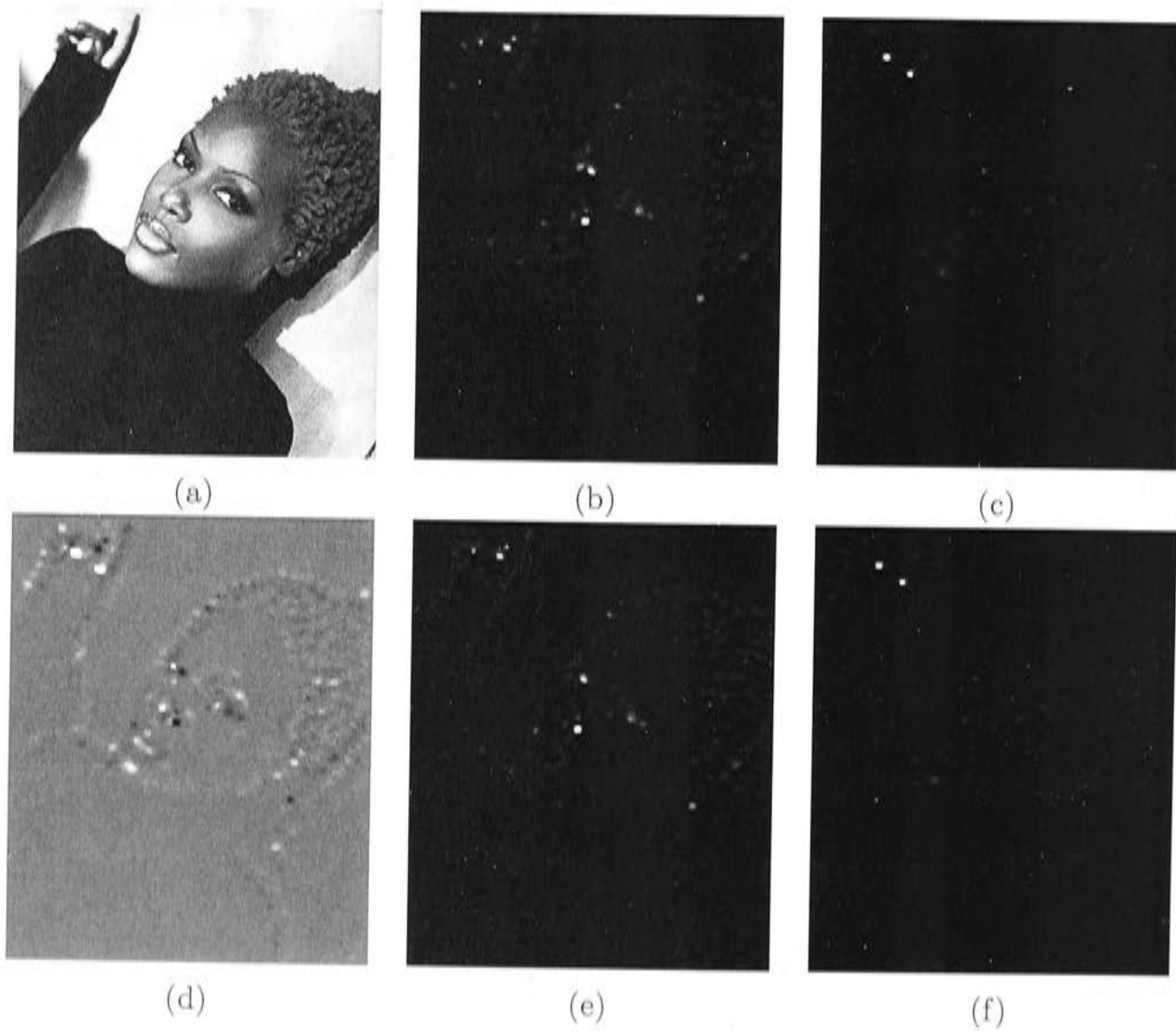


Figure 3.9: Examples of dark and bright symmetries. (a) Input image. (b) Dark symmetry. (c) Bright symmetry. (d) Dark and bright symmetry image S . (e) Dark and (f) bright symmetry images obtained from thresholding S .

cel each other out as they do when calculating both dark and light symmetries together. However, experimentation has shown that although this alternative

approach gives slightly different dark/bright symmetry outputs the result is no better than simply counting only dark/bright affected pixels. Figure 3.9 shows examples of dark and bright symmetry determined using each approach. Since it is not necessary to cancel out inconsistent dark/light values, the first method is preferred as it offers a reduction in computation.

Dark symmetry is especially useful for detecting facial features that typically appear darker than the surrounding skin, more examples of dark symmetry are shown in Section 3.5.

3.3.3 Choosing a Constant \mathbf{A}_n

A faster implementation of the transform can be achieved by choosing the Gaussian kernel to be constant over all radii. The saving in computation is achieved by avoiding performing convolutions with \mathbf{A}_n for each radius. Choosing \mathbf{A}_n to be a fixed Gaussian still disperses the influence of the affected pixels, and can produce reasonable results. In this case only one convolution need be performed, and Equation 3.4 reduces to

$$\mathbf{S}' = \mathbf{G}_\sigma * \sum_n \mathbf{F}_n \quad (3.5)$$

where \mathbf{G}_σ is a 2D Gaussian with standard deviation σ .

3.4 A General Set of Parameters

As discussed in Sections 3.2 and 3.3 there are a number of different parameters and refinements to the basic transform. In Table 3.2 three general parameter sets suitable for different applications of the transform are presented. The *Full* setting is the best choice when the transform is to be applied in an unsupervised manner, it provides more detail at the expense of requiring more computation than the alternative settings. The *Fast* setting detects both bright and dark symmetry quickly, and the *Fast Dark* setting finds only regions of dark symmetry. The performance of each of these settings is presented in Section 3.5.

Table 3.2: Parameter Settings used for Experimentation

Parameter	Setting		
	Full	Fast	Fast Dark
Set of radii N	$\{n : n = 1, 2, \dots, 6\}$	$\{n : n = 1, 3, 5\}$	$\{n : n = 1, 3, 5\}$
Gaussian kernel			
Size	n	n	n
Standard deviation	$0.5n$	$0.5n$	$0.5n$
Radial strictness α	2	2	2
Small gradients ignored	0	2% ignored	2% ignored
Dark symmetry	Yes	Yes	Yes
Bright symmetry	Yes	Yes	No

Figure 3.10: 256×256 lena image (USC-SIPI Image Database).

3.5 Performance Evaluation

This section demonstrates the performance of the FRST on a range of images, and compares it with several prominent transforms from the reported literature.

3.5.1 Performance of the FRST

The FRST was applied to the standard 256×256 lena image (Figure 3.10) using both the *full* and *fast dark* parameter settings in Table 3.2. These results are presented as layers in Figure 3.11 with the corresponding peaks in dark radial symmetry indicated. The eyes both stand out, as do two other points in the image. Examining the non-eye peaks show that they both correspond to small roughly round dark regions in the original image, so it is unsurprising that the transform returns high values at these locations. It is interesting to note that the result with the full parameter setting detects the whites of the eyes as points

of light symmetry. The whites of the eyes are not always as distinctly visible as they are in this image, especially in video images or instances when the eyes are in shadow, however, in high quality images when the eye whites are visible the co-occurrence of light and dark symmetry in close proximity is a strong cue for eye detection.

Figure 3.12 demonstrates the performance of the transform on faces and other images. These figures were generated using the parameter settings presented in Table 3.2, and show how the transform can provide a useful cue for the location of facial features — especially eyes — in face images, as well as highlighting generic points of interest that are characterized by high contrast and radial symmetry. Note that the orientation-based symmetry is more sensitive to low-contrast features and texture. This sensitivity can be reduced by using a higher gradient threshold, however, such sensitivity is desirable when considering low contrast features such as the shadowed side of the face in Figure 3.16.

The intuitive notion that facial features are generic points of interest provides a useful benchmark for evaluating point of interest operators. Whilst the application of these operators is by no means limited to facial feature detection (Chella *et al.*, 1999; Di Gesù and Valenti, 1995a; Minor and Sklansky, 1981) this is certainly the most common application area (Di Gesù and Valenti, 1995a; Intrator *et al.*, 1995; Lin and Lin, 1996; Reisfeld *et al.*, 1995; Reisfeld and Yeshurun, 1998; Sela and Levine, 1997; Sun *et al.*, 1998). Facial images provide a useful case study, offering images of widely varying appearances with well defined sets of interest points, as well as directly addressing the primary application area of point of interest detectors.

The FRST has been implemented in a realtime vision system. The realtime code was written in C++ and made use of the Intel Image Processing Primitives (version 2.05) to achieve a mean processing time of 13.2 ms (standard deviation of 0.08 ms) per 240×320 image frame, on a 1.4 MHz Pentium III running under Linux. The realtime system detects orientation-based symmetry online using the *fast dark* settings detailed in Table 3.2. However, to increase efficiency, uniform square kernels were used rather than Gaussians to blur the response at each radius (Equation 3.1). Figure 3.13 shows some snap shots of the system output. The results highlight the eyes and mouth of the subjects well, and there are virtually no noticeable artifacts caused by using uniform square kernels rather than Gaussians.

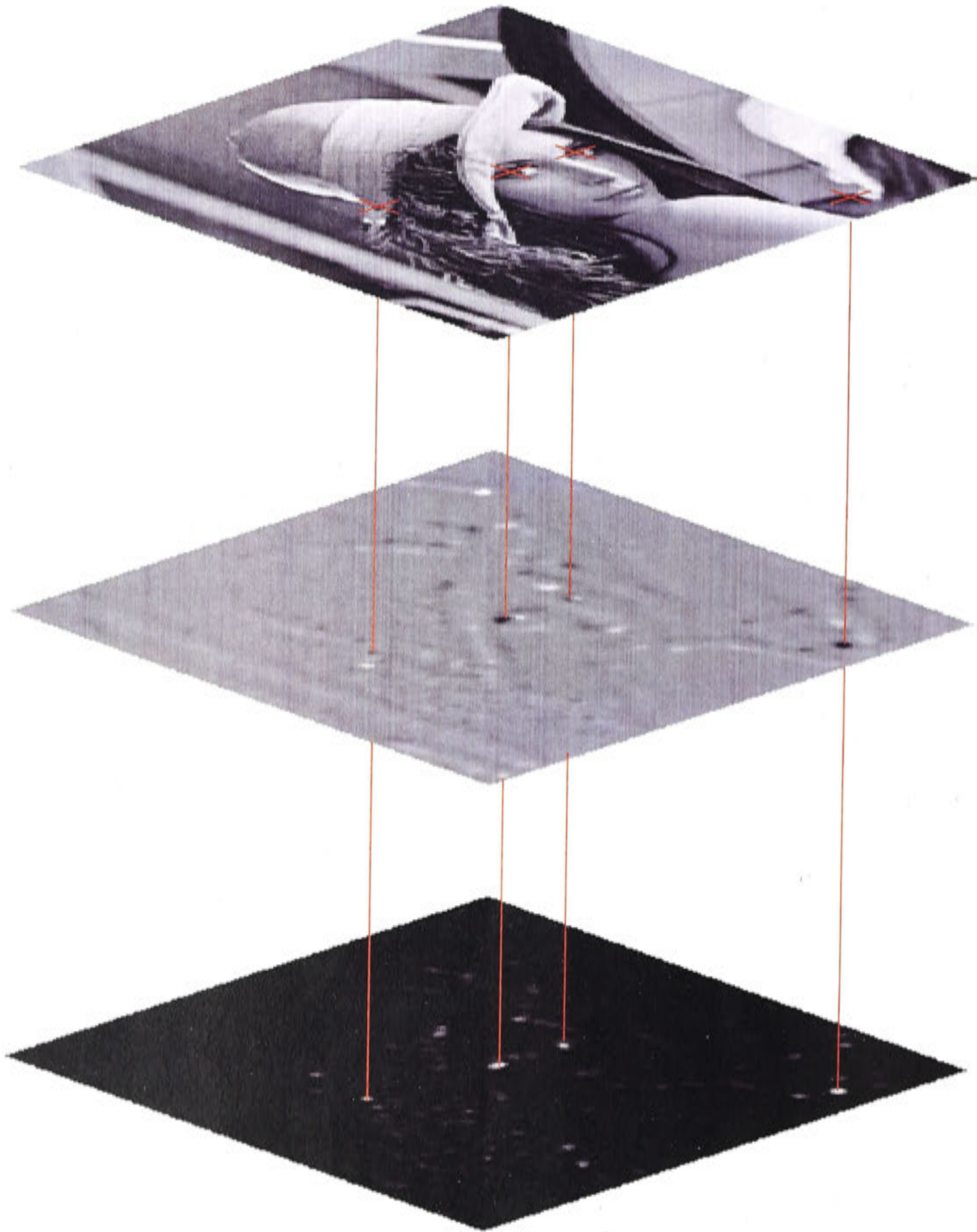


Figure 3.11: Results of applying the FRST to the 256×256 lena image, with corresponding points of high dark radial symmetry indicated. Top: original image. Middle: result with the *full* parameter setting. Bottom: result with the *fast dark* parameter setting.

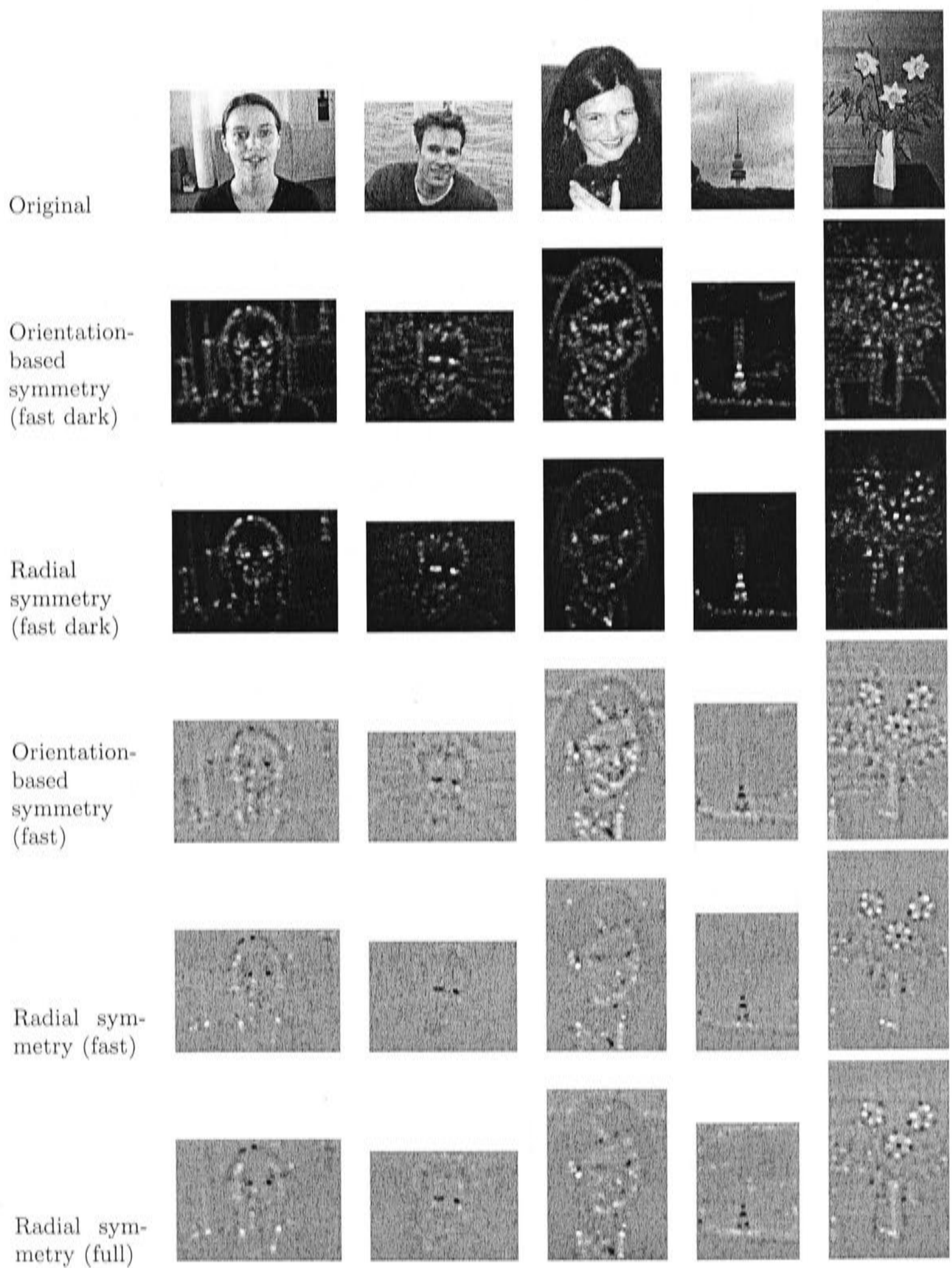


Figure 3.12: The FRST applied to face and other images. The form of the transform and the parameter settings used for each row are indicated on the left. The left most image is from the BioID Face Database Research (2001) and has been sub-sampled to half its original size.

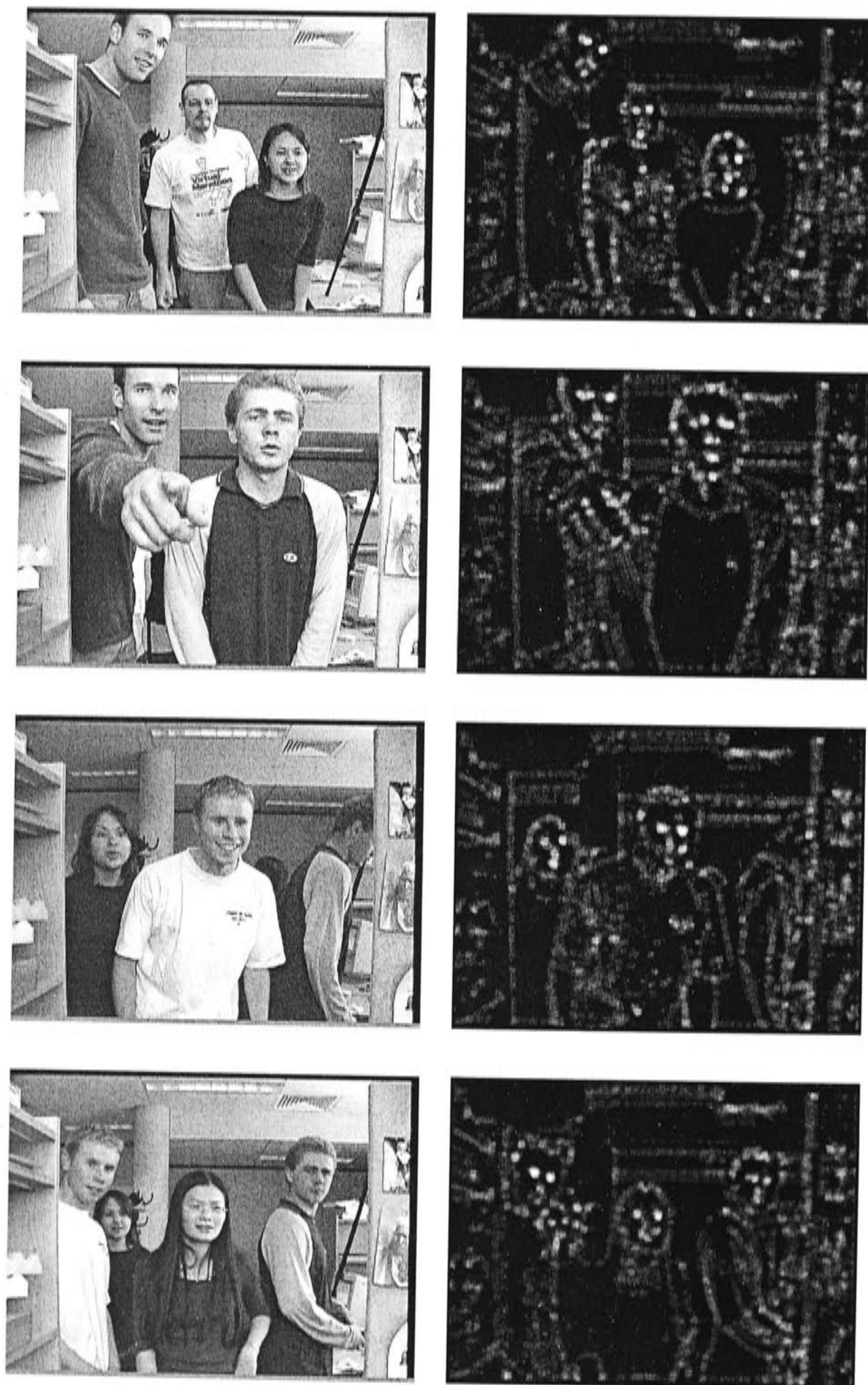


Figure 3.13: The orientation-based *fast dark* implementation of the FRST being calculated online in realtime. The left column shows sample input images and the right column shows the output, all images are 240×320 pixels.

The realtime code was also tested off-line on the 256×256 image in Figure 3.10 and timed over 10,000 iterations to determine a mean processing time of 10.8 ms for the calculation of the *fast dark* orientation-based symmetry on this image.

3.5.2 Comparison with Existing Transforms




In Chapter 2 we reviewed a number of existing methods for calculating radial symmetry (see Section 2.1.5). We now compare the performance of the FRST against the more prominent of these existing transforms, namely:

- Sela and Levine's realtime attention mechanism (Sela and Levine, 1997),
- Reisfeld's generalized symmetry transform for both dark and radial generalized symmetry (Reisfeld *et al.*, 1995),
- Kovese's symmetry from phase (Kovese, 1997),
- Di Gesù *et al.*'s discrete symmetry transform (Di Gesù and Valenti, 1995a), and
- Minor and Sklansky's implementation of the Circular Hough transform (Minor and Sklansky, 1981).

Kovese's symmetry from phase was calculated for 6 filter orientations and 4 scales ranging from 2 to 2^4 pixels in diameter. All other methods were implemented with a local neighbourhood radius of 6 pixels, allowing local symmetry to be detected in a neighbourhood of up to 13×13 pixels about each point. Where necessary the gradient orientation was quantized into 8 bins.

Each of the transforms was implemented in Matlab 5.3 (Kovese's symmetry from phase was implemented using Kovese's own Matlab code (Kovese, 1999b)) and the output computed. For the majority of the transforms an estimate of the approximate number of floating point operations involved was obtained from Matlab, however, for Di Gesù *et al.*'s discrete symmetry transform and Sela and Levine's realtime attention mechanism this was not the case. These transforms involve optimized low-level processes that were not practical to emulate in Matlab, so the number of operations required is not reported here. (Unsurprisingly, the *non-optimized* implementations used to generate the visual results shown required computation well in excess of the other methods.)

Table 3.3: Estimated Computation Required for Different Transforms

Transform	Computations (Mflop)		
	 Figure 3.14	 Figure 3.15	 Figure 3.16
FRST			
Radial Symmetry			
Full	17.9	18.9	23.4
Fast	8.06*	7.26*	8.51*
Fast Dark	6.76*	5.99*	7.4*
Orientation Symmetry			
Fast	7.44*	6.69*	7.87*
Fast Dark	6.45*	5.71*	7.09*
Existing Transforms			
Reisfeld <i>et al.</i> 's Generalized Symmetry			
Radial	300	259	349
Dark	207	179	239
Minor and Sklansky's Circular Hough	30	33.2	43.1
Kovesi's Symmetry from Phase	601	196	912

* Note that the Fast and Fast Dark parameter settings ignore small gradients and are not calculated across all radii (see Table 3.2).

The results are shown in Figures 3.14 to 3.16 and the computations required are presented in Table 3.3. These results demonstrate that the FRST can provide comparable or superior results to existing techniques whilst requiring a relatively low level of computation. As noted in the footnote to Table 3.3 the *fast* and *fast dark* parameter settings ignore small gradients and are not calculated across all radii, however, the transform is still able to provide useful results, and the computational efficiency is increased. Other transforms may also benefit from these technique. Indeed, Reisfeld initially considered ignoring small gradients (Reisfeld, 1993). However, the effect of these variables on other transforms has not been explored in this thesis.

The realtime attention mechanism of Sela and Levine (1997) provides cloud-like approximations of interest points. The final step of this transform involves identifying local maxima in this output as points of interest. These have been

marked with crosses in Figures 3.14 to 3.16 with the size of the cross corresponding to the value of the transform at the local maximum. The transform detects both eyes in the face in Figure 3.15 and the high contrast eye in Figure 3.16, but it also awards high interest to non-radially symmetric edges and areas of texture, and fails to detect the circular wheels of the car in Figure 3.14.

The results from the generalized symmetry transform show good detection of regions of interest by generalized radial symmetry, however, the generalized dark symmetry tends to highlight edges in addition to points of interest. The high computational load of the generalized symmetry transform (and other methods that consider symmetry in a local neighbourhood about each pixel (Lin and Lin, 1996)), comes from the computational load scaling with the square of the radius of the neighbourhood. The larger the neighbourhood the more pixels that must be considered when calculating the transform at each point in the image. Even for modest sized neighbourhoods, such as the 13×13 pixel neighbourhood used for the experimentation on the 320×240 , 256×256 and 256×341 images in Figures 3.14 to 3.16, the computation is considerable.

Calculating the symmetry from phase detects areas of high bilateral or radial symmetry independently of contrast. This method is not designed to detect points of interest in scenes, however, it provides a detailed map of the underlying symmetries present across the image that is instructive to consider in relation to other “symmetry operators”. Comparing the results from this transform with those of Reisfeld’s generalized *dark* symmetry we see that (as noted by Kovesi (1997)) the latter is essentially a combined measure of the underlying symmetry *and* the contrast. Furthermore, comparing the lines of bilateral symmetry (from the phase symmetry image) with the points of high radial symmetry from Reisfeld *et. al.*’s generalized *radial* symmetry, confirms that radial, rather than bilateral symmetry is a better detector of points of interest in Figures 3.14 to 3.16.

The discrete symmetry transform tends to highlight either side of high contrast lines, with the result that when such a line forms a ring, such as the wheels of the sports car in Figure 3.14, it is strongly highlighted. However, there is also a lot of bold highlighting of non-circular edges and regions of high texture that do not exhibit radially symmetry. While detecting these features may be desirable for some applications, they distract from the emphasis placed on radially symmetric points, and detract from the performance of the transform as a symmetry-based interest detector.

Minor and Skalansky's implementation of the Circular Hough transform comes closest to rivalling the computational efficiency of the FRST, yet it provides only four levels of output. It was designed for detecting dark blobs in infrared images, and when applied as a point of interest detector to the photographs shown here it detects many other points in addition to the primary interest points. In particular it returns high values along edges, such as the frame of the mirror in Figure 3.15, and is easily confused by textured surfaces, such as the grass in Figure 3.14.

Table 3.4 lists the order of computation required to compute the transforms on an image of K pixels, where local symmetry is considered in an $N \times N$ neighbourhood, and for those methods which require gradient quantization the gradient is quantized into B bins. The complexity $O(KN)$ of the FRST is lower than all other transforms considered, with the exception of Di Gesù *et al.*'s discrete symmetry transform that has complexity $O(KN)$ or $O(KB)$. When calculating the discrete symmetry transform with complexity $O(KB)$ (Di Gesù and Palenichka, 2001) it is essential to calculate it across four or more angular bins, whereas when calculating the FRST it is not necessary to compute it at all radii $1 \dots N$ (see Section 3.2.1). Likewise the order $O(KN)$ implementation of the discrete symmetry transform (Palenichka *et al.*, 2001) can be calculated at only a subset of the radii. However, the results from the discrete symmetry transform are quite different from the method presented in this thesis, with edges and areas of high texture, in addition to points of radial symmetry, typically being awarded high responses.

Table 3.4: Computational Order of Different Transforms

Transform	Order
FRST	KN
Generalized Symmetry Transform Reinfeld <i>et al.</i>	KN^2
Gradient-based Inhibitory Mechanism (Lin and Lin, 1996)	KN^2
Discrete Symmetry Transform (Di Gesù and Palenichka, 2001; Palenichka <i>et al.</i> , 2001)	KB or KN
realtime Attentional Mechanism (Sela and Levine, 1997)	KBN
Circular Hough Transform Minor and Sklansky (1981)	KBN

The key to the speed of the FRST lies in the use of *affected pixels* to project the effect of gradient elements. This allows an approximation of the effect of each gradient element on the radial symmetry of the pixels around it, without specifically considering neighbourhoods about each point like Lin and Lin (1996) and Reinfeld *et al.* (1995), or requiring multiple calculations for different gradient

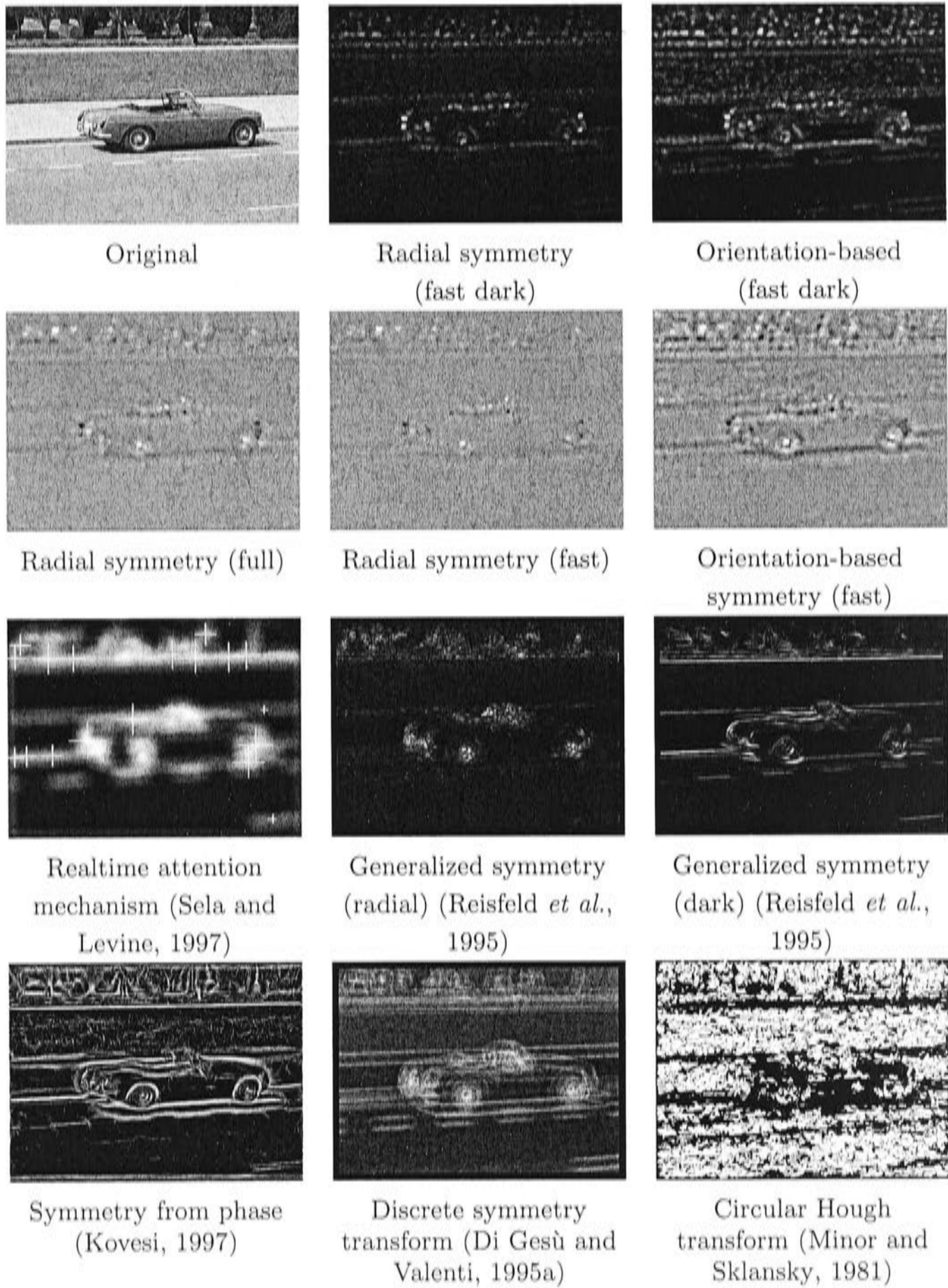


Figure 3.14: Comparison of performance on a 320×240 outdoor image. The top two rows show the performance of the FRST, the bottom two rows show the output from other available transforms.

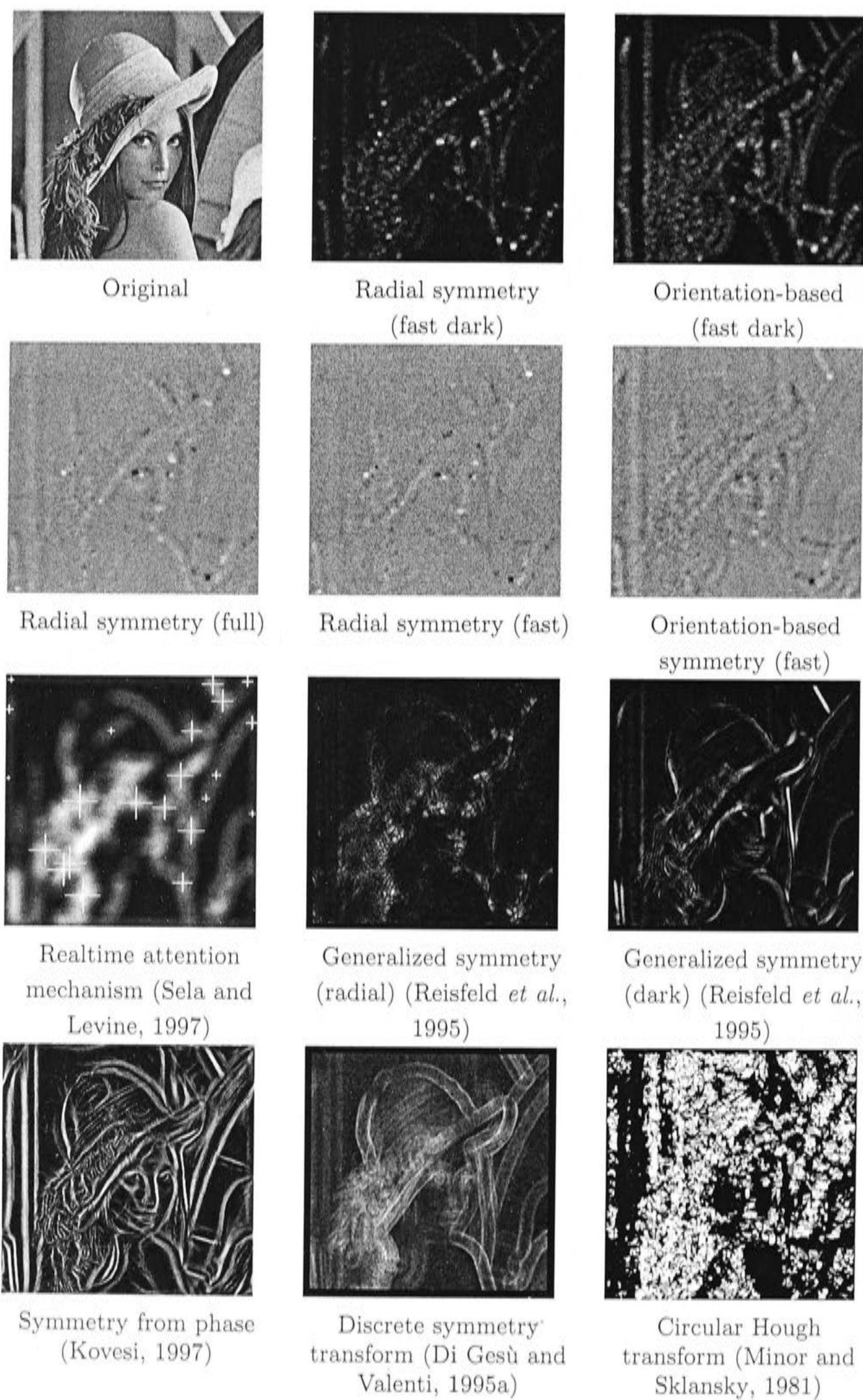


Figure 3.15: Comparison of performance on the standard 256×256 lena image. The top two rows show the performance of the FRST, the bottom two rows show the output from other available transforms.

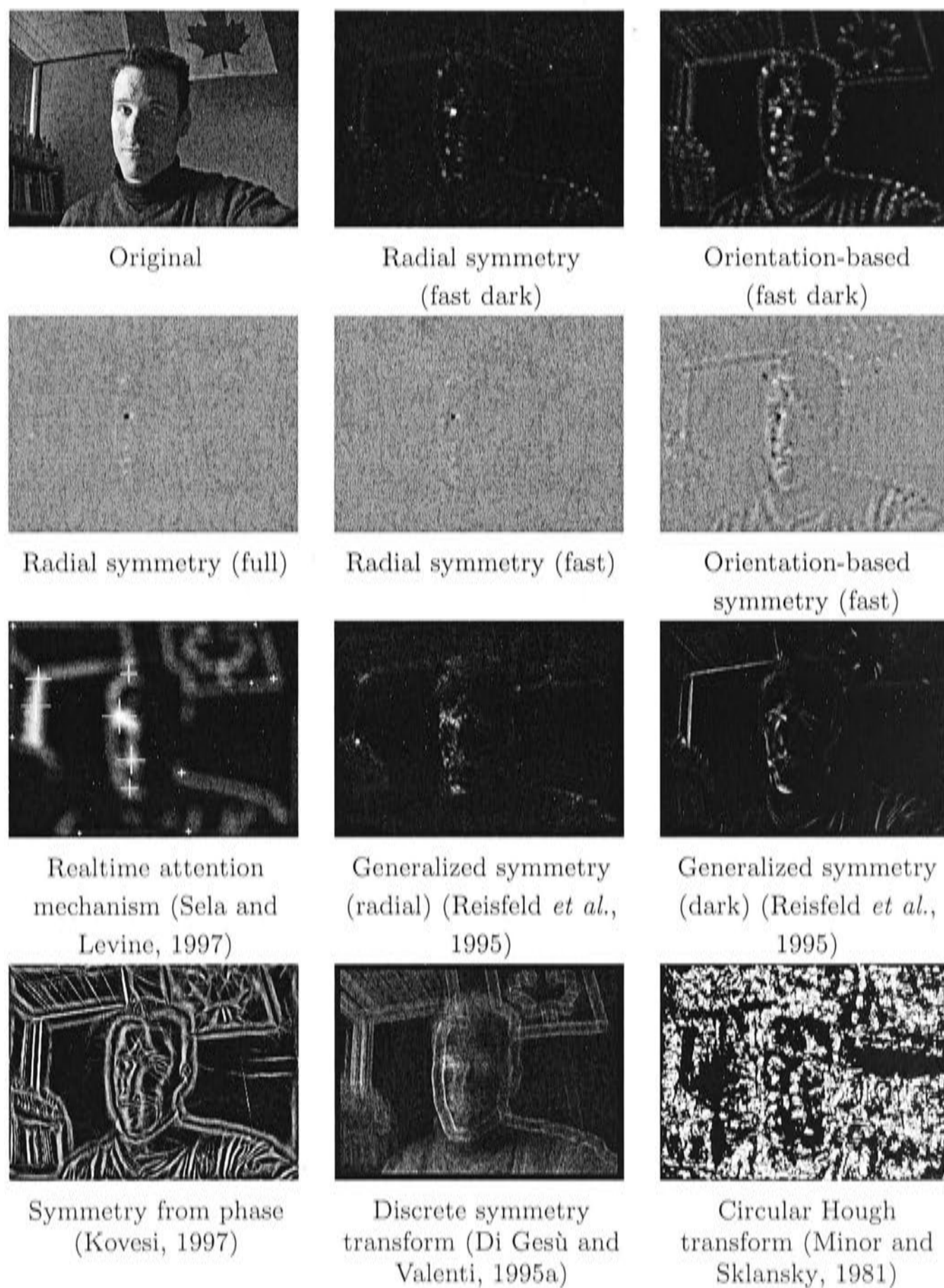


Figure 3.16: Comparison of performance on a 256×341 image of a face in half shadow. The top two rows show the performance of the FRST, the bottom two rows show the output from other available transforms.

orientations, as do many other methods (Di Gesù and Valenti, 1995a; Kovese, 1997; Minor and Sklansky, 1981; Sela and Levine, 1997).

Unlike other transforms the fast symmetry transform differentiates between dark and bright regions of radial symmetry, while allowing both to be computed simultaneously. Alternatively just dark (or bright) points of symmetry can be considered exclusively with an associated reduction in computation.

3.6 Summary

In this chapter we have presented a new transform, the Fast Radial Symmetry Transform (FRST), that utilizes local radial symmetry to highlight points of interest within a scene. Its low computational complexity and fast run-times make this method well suited for realtime vision applications. The performance of the transform has been demonstrated on a variety of images and compared with leading techniques from the literature. Both as a facial feature detector and as a generic region of interest detector the FRST is seen to offer equal or superior performance to contemporary techniques at a relatively low computational cost. A realtime implementation of the transform was also presented and shown to be an effective cue for highlighting peoples eyes as they moved in front of the camera.

This transform provides a means of efficiently detecting the presence of radial symmetry for realtime applications, in particular facial feature detection. In Chapter 4 we use this transform as one of the key visual cues in our face localisation system, and in Chapter 5 we applied it to preprocessing images for face registration.

Chapter 4

Face Localisation

THE first step to enabling a computer to see a person's face is to localise and track the approximate location of the face in an image sequence. We refer to this as *face localisation* which is the topic of this chapter.

Visually acquiring and tracking faces and other targets is a key problem in computer vision, and new and innovative techniques are constantly being developed. However, despite the impressive results obtained, it is clear that no single cue can perform reliably in all situations. The key to an efficient and robust vision system for tracking is to intelligently combine information from a number of different cues, whilst effectively managing the available computational resources.

The development of such a system must address several issues: which cue(s) should be used and when, how should the cues be combined, and how much computational resource should be expended on each cue.

This chapter presents a framework for a vision system that addresses these issues by fulfilling the following criteria:

- efficiently allocate finite computational resources when calculating cues, accounting for the cue's expected utility and resource requirement,
- facilitate cues running at different frequencies,
- locate a target in multi-dimensional state space, eg. determining the target's 3D location and orientation, and
- allow tracking of multiple hypotheses.

Firstly we introduce some background theory in Section 4.1 which forms the basis for our system. The overall architecture of the system is described in Section 4.2. Section 4.3 describes an implementation of the system that locates and tracks a person's head in a cluttered environment. Visual cues are discussed and experimental results are presented. Section 4.4 demonstrates how the system can be extended to track multiple targets. Section 4.5 closes with a summary of the key points.

4.1 A Bayesian Approach to Target Localisation

The system uses Bayesian probability theory to fuse information from different time instances and sensing modalities. A particle filter is used to approximate the resulting PDF and maintain multiple hypotheses of the target location. This section details the Bayesian framework which leads to Markov localisation. Particle filtering is also discussed, and it is shown how a particle filter can be applied to model the probability density resulting from Markov localisation.

4.1.1 Markov Localisation

Given a state space of possible target poses, the problem of target localisation can be expressed probabilistically as the estimation of the posterior probability density function over the space of possible poses, based on the available data. That is, at time t estimate the posterior probability $P(s_t|e_{0..t})$ of a state s_t given all available evidence $e_{0..t}$ from time 0 to t .

Using Bayesian probability theory and applying the *Markov assumption*¹ the desired probability $P(s_t|e_{0..t})$ can be expressed recursively in terms of the current evidence and knowledge of the previous states. This is referred to as *Markov Localisation*, and is commonly used in mobile robotics. It is represented mathematically by the following equation,

$$P(s_t|e_{0..t}) = \eta_t P(e_t|s_t) \int P(s_t|s_{t-1}) P(s_{t-1}|e_{0..t-1}) ds_{t-1} \quad (4.1)$$

¹The *Markov assumption* states that the past is independent on the future given the current state.

where η_t is a constant normaliser that ensures the probabilities sum to one, $\eta_t = 1/P(e_t|e_{0..t-1})$.

The derivation, as outlined by Thrun (2000), sequentially applies Bayes' rule, the Markov assumption, the theorem of total probability and the Markov assumption again, and is detailed below.

Firstly Bayes' rule is applied, one of the probabilities is expanded and Bayes' rule is applied again.

$$\begin{aligned}
 P(s_t|e_{0..t}) &= \frac{P(e_{0..t}|s_t)P(s_t)}{P(e_{0..t})} \\
 &= \frac{P(e_t|e_{0..t-1}, s_t)P(e_{0..t-1}|s_t)P(s_t)}{P(e_{0..t})} \\
 &= \frac{P(e_t|e_{0..t-1}, s_t)P(s_t|e_{0..t-1})P(e_{0..t-1})P(s_t)}{P(s_t)P(e_{0..t-1})P(e_t|e_{0..t-1})} \\
 &= \eta_t P(e_t|e_{0..t-1}, s_t)P(s_t|e_{0..t-1})
 \end{aligned}$$

Applying the Markov assumption allows $P(e_t|e_{0..t-1}, s_t)$ to be rewritten as $P(e_t|s_t)$ giving

$$P(s_t|e_{0..t}) = \eta_t P(e_t|s_t)P(s_t|e_{0..t-1})$$

Since the set of all possible states at a given time represents a collection of mutually exclusive events whose probabilities sum to one, the theorem of total probability enables $P(s_t|e_{0..t-1})$ to be written as a sum over all possible state values at time $t - 1$ as follows

$$P(s_t|e_{0..t}) = \eta_t P(e_t|s_t) \int P(s_t|e_{0..t-1}, s_{t-1})P(s_{t-1}|e_{0..t-1})ds_{t-1}$$

The Markov assumption can then be applied again to simplify $P(s_t|e_{0..t-1}, s_{t-1})$ leading to the formulation presented in Equation 4.1. This formulation provides

a recursive means of estimating the probability of the current state given all the evidence sighted since sensing began.

4.1.2 Markov Localisation with a Particle Filter

A particle filter is applied to model the distribution in Equation 4.1. In Chapter 2 we explained how a particle filter is able to locate targets in state space while only measuring the probabilities of target hypotheses at a discrete number of locations. The particle filter effectively approximates the continuous distribution by a set of discrete samples. Thus we are only required to determine $P(s_t|e_{0..t})$ for a number of *discrete* values of s_t (and s_{t-1}), so Equation 4.1 becomes

$$P(s_t|e_{0..t}) = \eta_t P(e_t|s_t) \sum_i P(s_t|s_{t-1}^{(i)}) P(s_{t-1}^{(i)}|e_{0..t-1})$$

where (i) denotes the i^{th} discrete value (i.e., particle).

There are two main parts to the right hand side of Equation 4.2, the probability $P(e_t|s_t)$ and the summation.

Each term of the summation describes the probability of a specific particle $s_{t-1}^{(i)}$ migrating to the location s_t in the next time step. Summing over all the particles in the previous time frame gives the probability of a particle occurring at a particular location in state space s_t , given the locations and probabilities of all the particles at time $t - 1$. To see how this is modelled by the particle filter we need to consider the constituents of each term of the summation, namely the probabilities $P(s_t|s_{t-1}^{(i)})$ and $P(s_{t-1}^{(i)}|e_{0..t-1})$. The second of these is modelled by re-sampling with replacement (step 2 in Figure 4.1), where this probability is the chance of re-sampling a particle. The other probability $P(s_t|s_{t-1}^{(i)})$ is modelled by Brownian motion and deterministic drift (steps 3 and 4 in Figure 4.1), that determine the location of a particle s_t given its previous location s_{t-1} .

The other part of Equation 4.2, $P(e_t|s_t)$, is measured from sensor information. In our case this amounts to examining the visual input at the appropriate location in the image and estimating the probability that this location contains the target. This is modelled by the update PDF phase in the particle filter (step 1 in Figure 4.1). The sensing process is described in more detail in Section 4.2.2.

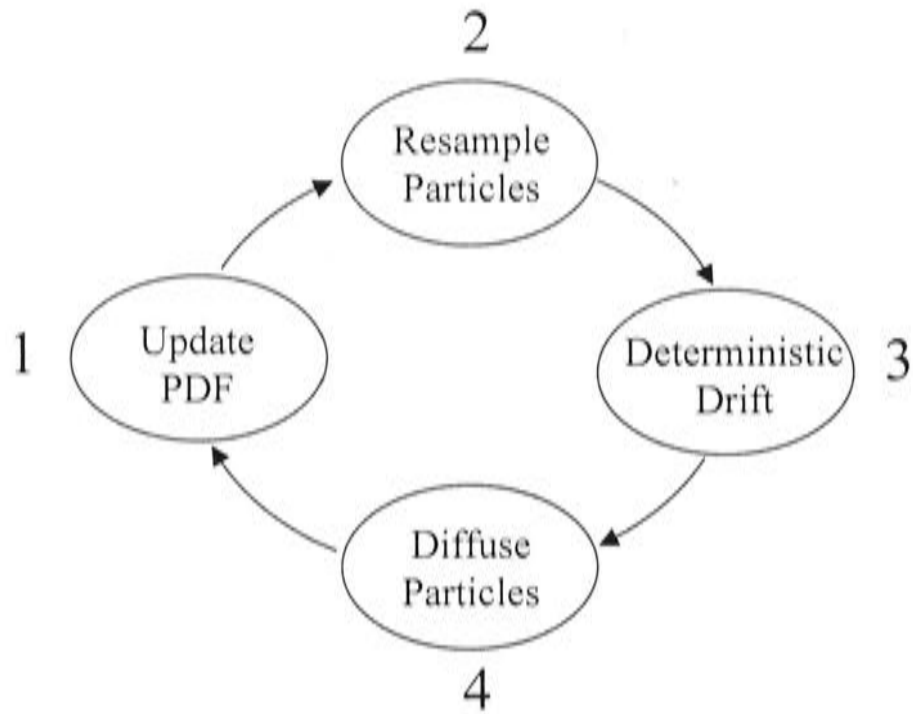


Figure 4.1: The four steps of the particle filter.

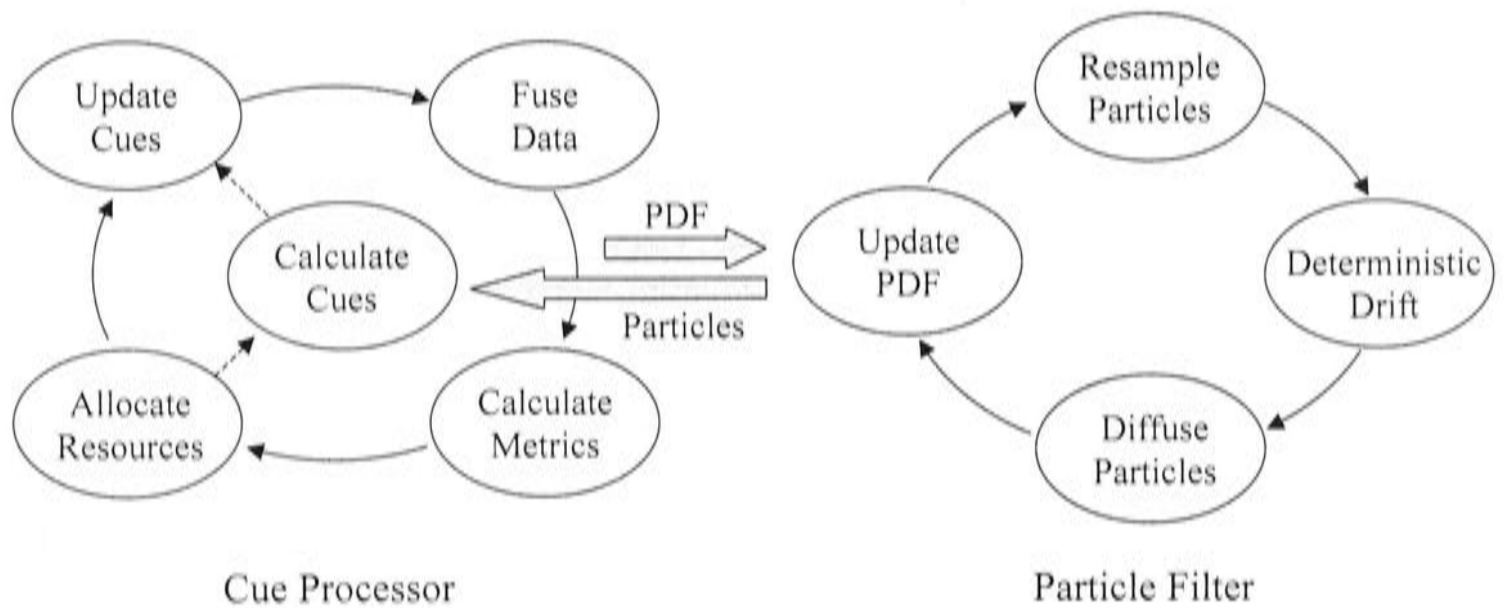


Figure 4.2: System architecture.

4.2 System Design

The system detailed in this chapter uses a particle filter to track a population of target hypotheses in state space. A number of cues are calculated from image and state information and combined to provide evidence strengthening or attenuating the belief in each hypothesis.

Figure 4.2 shows the structure of the system. It consists of two subsystems: a particle filter and a cue processor, each of which cycle through their loops once per frame. These subsystems interact as shown by the thick arrows in the figure. The particle filter passes the current particle locations to the cue processor. The cue processor determines the probabilities for the particles and passes these back to the particle filter. Each of these subsystems is discussed in further detail below.

4.2.1 Particle Filter

A description of particle filtering and the mechanisms driving the process has been given in Chapter 2 Section 2.2. In this chapter we apply a particle filter to track a target's location in 3D space. To summarise this process Figure 4.3 shows a series of schematics of a particle filter tracking a person's head, where the state variables are the 3D position of a person's head (x, y, z) , and its orientation θ about the optical axis. Initially the hypotheses are distributed uniformly about the state space. Next, the probability that each hypothesis represents the true target location is determined by examining the appropriate image location, and thus a probability value is determined for each hypothesis. The hypotheses are re-sampled, and subjected to deterministic drift and diffusion. This process is repeated for every new image frame in the sequence and results in clustering of hypotheses around the most promising target locations.

The primary appeals of the particle filter approach to localisation and tracking are its scalability (computational requirement varies linearly with the number of particles), and its ability to deal with multiple hypotheses and thus more readily recover from tracking errors. However, the particle filter was applied here for several additional reasons:

- it provides an efficient means of searching for a target in a multi-dimensional state space.
- it reduces the search problem to a verification problem, i.e., is a given hypothesis face-like according to the sensor information?
- it allows fusion of cues running at different frequencies.

The last point is especially important for a system operating multiple cues with limited computational resources, as it facilitates running some cues slower than frame rate (with minimal computational expense) and incorporating the result from these cues when they become available.

If a cue takes n frames to return a result, by the time the cue is ready, the particles will have moved from where they were n frames ago. To facilitate such cues the system keeps a record of every particle's history over a specified number of frames k . The cue value determined for a particle $n \leq k$ frames ago can then be assigned to the children of that particle in the current frame,

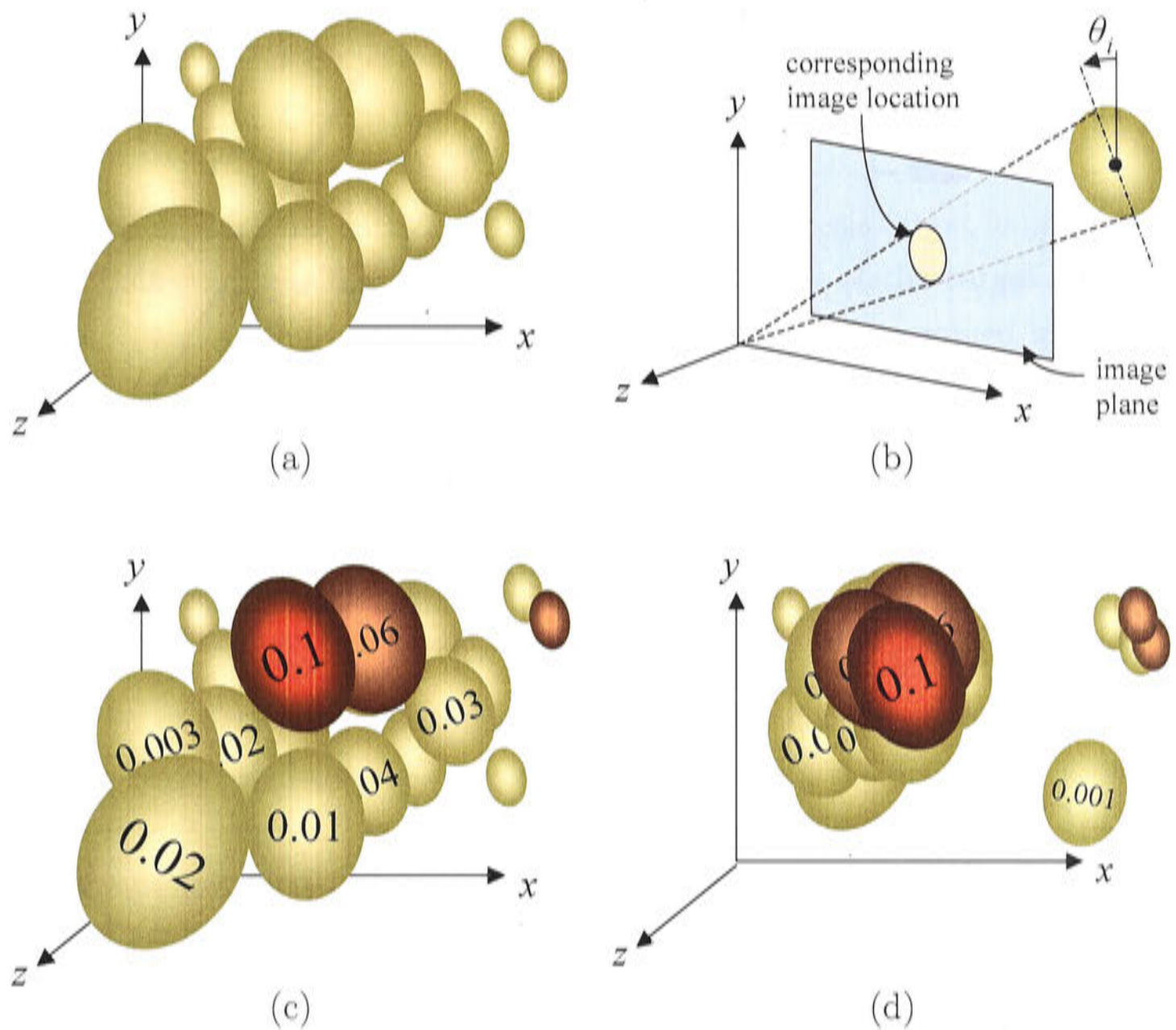


Figure 4.3: Schematics of particle filter tracking a head in (x, y, z, θ) state space. (a) Initial hypotheses are uniformly distributed across state space. (b) The probability of each hypothesis being the target is measured from the image. (c) These probabilities are assigned to each hypothesis. (d) Over time the hypotheses converge to the most “target-like” locations.

thus propagating forward the cue’s response to the current frame. Conversely, probabilities associated with particles that were not propagated are discarded. Figure 4.4 shows a simplified example with four particles in a one-dimensional state space, showing the evolution of this population over four time steps. The blue and orange particles at $s_i(t)$ are children of the same coloured particles from $t - 4$, thus a slow cue that takes four frames to compute is calculated for the particles $s_i(t - 4)$ and the values for the blue and orange particles assigned to the respective children of these particles in frame t , whilst values calculated for the green and red particles are discarded.

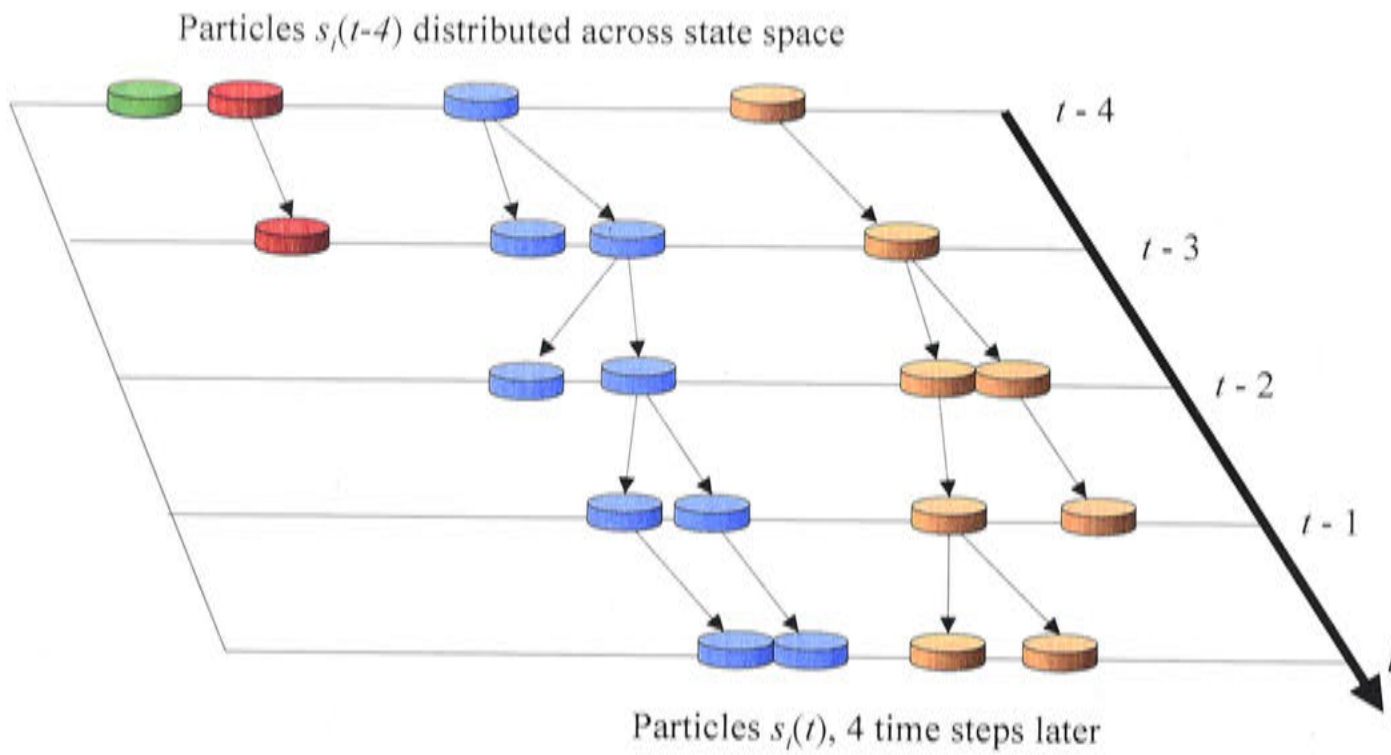


Figure 4.4: Example of particle population evolving over time, showing the history.

4.2.2 Cue Processor

Whilst the particle filter maintains a record of target hypotheses and propagates these through in state space, it is the *cue processor* that deals with the calculation and fusion of cues necessary to effectively measure the probability of each of these hypothesis. The cue processor also determines metrics measuring the performance of each cue, and the allocation of computational resources to individual cues.

Each frame the cue processor cycles through the steps illustrated in Figure 4.2:

1. Update cues: accesses recently calculated cues.
2. Fuse data: fuses the results of different cues to estimate the overall probability for each hypothesised target state.
3. Calculate metrics: determine the metrics for each cue that quantify how well that cue performed on the last image frame.
4. Allocate resources: based on the anticipated performance of the individual cues, allocate computational resources to maximise the quality of information obtained.

The *calculate cues* component of the system accepts requests for cue measurements and handles the requests using only the quantity of computational resources

allocated to it by the *allocate resources* component.

Calculating and Updating Cues

Each particle from the particle filter presents a hypothesis target location in state space. Using a pinhole camera model and knowledge of the target dimensions, the size, location and orientation of each hypothesis is determined in the image, as shown in Figure 4.3 (b).

Each cue returns a set of probabilities $\{P(e_t^{(i)}|s_t^{(j)})$ for $j = 1 \dots N\}$ indicating the i^{th} active cue's belief in the j^{th} hypothesis, where N is the total number of particles.

Calculating some cues may take longer than the time available between sequential frames. In this case the cue is not available to the update cue component in the following frame, and the cue will not be updated until the new value is ready. As discussed in Section 4.2.1, these slow cues are accommodated for by the update PDF component that is able to propagate their effect through to the probability values in the current frame.

The visual cues applied depend on the target being detected. In the face localisation implementation in Section 4.3 several simple cues are described for detecting a person's face in clutter. Section 4.4 describes cues for detecting faces and hands.

Fusing Cues

A crucial question when fusing sensor information is how to combine the probabilities obtained from different sensor modalities.

We assume the different cues are probabilistically independent. Whilst this assumption is not strictly true across all cues it is true in most cases, and it allows us to fuse the cues via simple multiplication of probabilities. However, there is a problem with zero probability values when fusing cues in this fashion, since multiplication with zero will always result in zero. For this reason we re-scale and offset the probabilities from zero by an amount $\alpha \in (0, 1)$. Subsequently, the probabilities from the cues are fused to determine the overall belief in the j^{th} hypothesis $P(e_t|s_t^{(j)})$ at time t as follows

$$P(e_t|s_t^{(j)}) = \prod_i (P(e_t^{(i)}|s_t^{(j)})(1 - \alpha) + \alpha)$$

In our system $\alpha = 0.1$ was used allowing cues with low responses to strongly attenuate the combined probability, whilst ensuring that a single cue returning zero will not force the combined probability to zero.

Quantifying Cue Performance

The performance, or *utility*, of each active cue is estimated every frame, and used to decide the distribution of computational resources across the cues.

Fusing the results of all available cues is assumed to give the best estimate of the true PDF $P(e_t|s_t)$ across the state space. So the performance of the j^{th} cue can be quantified by measuring how closely the cue's PDF $P(e_t^{(j)}|s_t)$ matches $P(e_t|s_t)$. This can be done using the relative entropy, or the Kullback-Leibler distance (Kullback and Leibler, 1951), an information theoretic measure of how accurate an approximation one PDF is to another, given by

$$\delta_t \left(P(e_t|s_t), P(e_t^{(j)}|s_t) \right) = \sum_i P(e_t|s_t^{(i)}) \log \frac{P(e_t|s_t^{(i)})}{P(e_t^{(j)}|s_t^{(i)})}$$

where s_t are the particle states at time t . Soto and Khosla (2001) used this metric to rate the performance of their cues, and Triesch and von der Malsburg (2000) considered it, but opted for a simpler *ad hoc* measure.

Our system uses this approach and we define the utility of the j^{th} cue at time t as

$$u_t(j) = \delta_t(P(e_t|s_t), P(e_t^{(j)}|s_t)) \quad (4.2)$$

Resource Allocation

A practical vision system has finite computational resources. To make the most of these resources it is important to use cues that provide the best quality information for the least computation. Our system is equipped with a range of visual cues. Different cues require varying amounts of computation and perform differently in different operational conditions.

The resource allocation component of our system aims to dynamically allocate computational resources to maximise the quality of information obtained per unit of computation. Quality of information is measured as the net *utility* of the cues computed, where the *utility* of each cue is determined from Equation 4.2.

This leads to a system that is able to adapt to changing operational conditions and adjust its use of cues accordingly. An additional advantage of this configuration is the flexibility it lends to changes in hardware and software, being able to readily accept new cues, sensing modalities, or changes in computational performance.

The system aims to locate and track targets, and give timely feedback regarding the target's location, so it is desirable to have at least some of the cues running at frame rate. For this reason a certain proportion of the time available for cue processing each frame is devoted exclusively to cues running at frame rate.

Slow cues were permitted to run once every 2, 4, or 8 frames. However, the longer a cue takes to generate information, the less useful that information is in terms of locating the target in the current frame. To account for this an exponential discount factor $d \in (0, 1)$ is introduced that attenuates the utility measure of a cue for each frame it is late. That is, the utility u is attenuated to $d^n u$ if it is n frames late.

For our system a simple resource allocation process was used that functions as follows:

- Allocate resources to cues running at frame rate:
 - Generate all combinations of cues that can be calculated in the time allocated for cues running at frame rate.
 - Choose the combination with the best overall utility.
- Allocate resources to cues running below frame rate:
 - Calculate the amount of time remaining for computing slow cues in the current frame (taking into account that some resources may already be allocated to slow cues that are still being computed from previous frames).
 - Determine all combinations of the remaining cues over all possible slower frame rates such that no combination exceeds the time available for the slower cues.
 - Calculate the net utility for each of these cue combinations using the discount factor to reduce the utility according to how late the cues are.
 - Choose the combination of slow cues offering the best overall utility.

This process is repeated for each new frame to continually reallocate the available resources throughout the system's operation.

4.3 Localising and Tracking a Head in a Complex Environment

As discussed in Chapter 1, face localisation is the first step to enabling a computer to see a face. It is also a precursor to numerous human computer interaction and surveillance tasks, such as tracking the head pose, attempting face recognition or expression recognition, and facial feature detection and tracking. Much research has focussed on detecting and tracking faces in both still images and image sequences (see Chapter 2 Section 2.2). However, the search for a robust face localising and tracking method is far from over. In this section we apply the vision system developed in this chapter to this problem. Our system is demonstrated localising and tracking a person's head in a cluttered environment, whilst dealing with changing head pose, occlusion and changing lighting conditions.

4.3.1 Implementation

An implementation of the system was developed as an object orientated algorithm in Matlab. To simulate realtime resource requirements the computational cost for each cue was estimated from the CPU time required.

Two uncalibrated colour stereo video cameras as were used as sensors. The images from these cameras undergo some preprocessing and are then passed to the cues where each target location hypothesis is tested by computing all active cues. Figure 4.5 shows the sensing process when all cues are active. Both the preprocessing and hypothesis testing are discussed below.

4.3.2 Preprocessing

Preprocessing is only performed *once* for each new set of images, whereas hypothesis testing requires one test for *every* target hypothesis generated by the particle filter. The preprocessing required for each frame is governed by the cues that are to be computed. These dependencies are illustrated by the network in Figure 4.5.

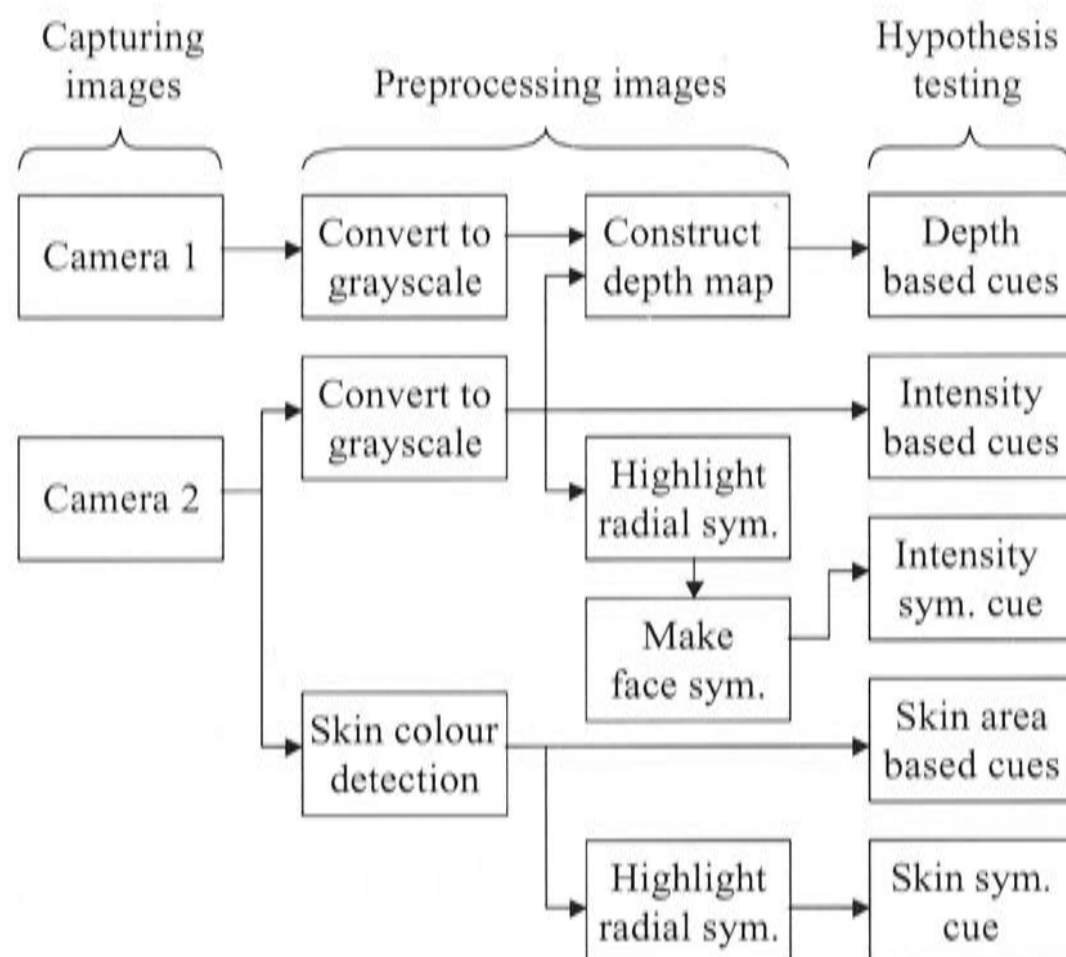


Figure 4.5: Sensing process

Figure 4.6 shows two colour 320×240 stereo images as received by the system's cameras, and the resulting outputs from preprocessing these images.

Depth Map

A dense depth map is generated from stereo intensity images using the approach of Kagami *et al.* (2000) (Chapter 2 Section 2.1.3). The optimised realtime implementation for our system was provided by Fletcher *et al.* (2001). For maximum efficiency pre-filtering was be done in software with a Difference of Gaussian filter, and stereo matching was performed using Sum of Absolute Differences. The resulting depth map presents the depths as viewed from Camera 2.

Skin Colour Detection

A skin colour likelihood image is generated from one channel of the stereo image stream, so that the value of each pixel in the skin colour likelihood image is indicative of the probability that there is skin colour at that location in the original image.

The skin colour likelihood of each pixel is determined by reference to a pre-

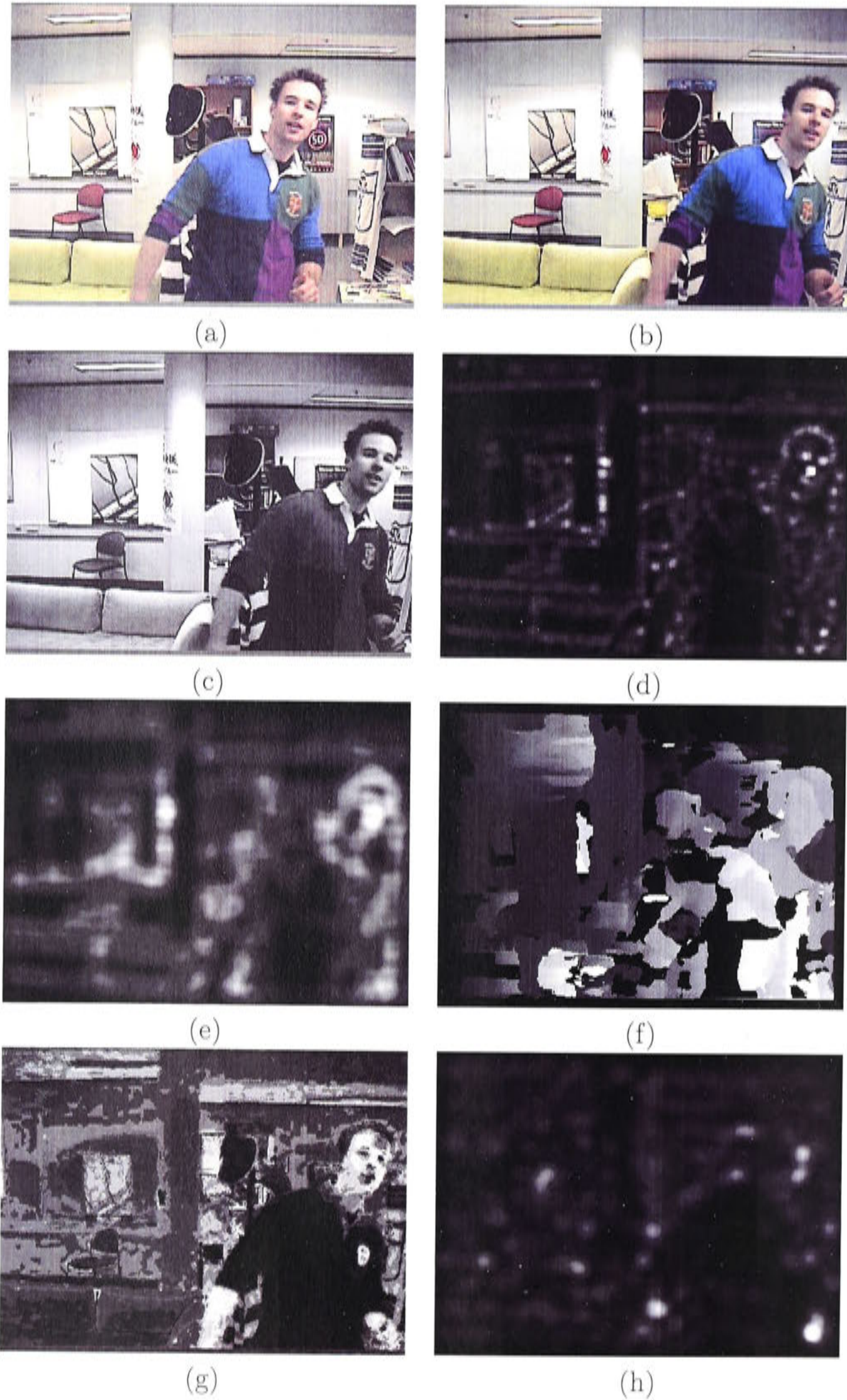


Figure 4.6: Preprocessing a colour stereo image pair. (a) Image from camera 1, (b) Image from camera 2, (c) Intensity image, (d) Radial symmetry image, (e) Facial symmetry image, (f) Depth map, (g) Skin colour likelihood image, (h) Radial symmetry of skin colour likelihood image searching for a radius of 15 pixels.

computed skin colour histogram. The histogram was generated using an extension of the method used by Cai and Goshtasby (1999) discussed in Chapter 2, where a two-dimensional histogram of skin chrominance was constructed in the *CIE ab* chrominance space. We use a three dimensional skin colour model in the YUV space built as follows: we discretise the colour space into $16 \times 64 \times 64$ bins (16 for Y and 64 for U and V), plot each skin colour sample in the discretised colour space, blur these by convolution with three dimensional Gaussians, and finally normalise the result so the maximum value is unity. 173,000 skin colour samples were used from 346 images of faces of people of varying race captured under different lighting conditions. Note: none of these samples were from people later tracked by the system.

Radial Symmetry

The new radial symmetry operator described in Chapter 3 was used to highlight possible eye locations in the original grey-scale image, and possible head locations in the skin colour likelihood image. The orientation-based variant of the transform is used in both cases.

When applied to the skin colour likelihood image the transform highlights light regions that are approximately circular and of a similar diameter to a face, see for example Figure 4.6 (h).

The operator was also applied to the intensity image to highlight small dark regions such as the eyes (Figure 4.6 (d)). This output is then convolved with a blurred annulus to highlight the regions between potential eye pairs. This second output is referred to, somewhat arbitrarily, as the *facial symmetry image* (Figure 4.6 (e)).

Each application of the radial symmetry operator is performed at three different radii to detect targets at three ranges of depth away from the camera.

4.3.3 Hypothesis Testing

As stated in Section 4.2.2 at time t each cue returns a set of probabilities

$$\{P(e_t^{(i)} | s_t^{(j)}) \text{ for } j = 1 \dots N\}$$

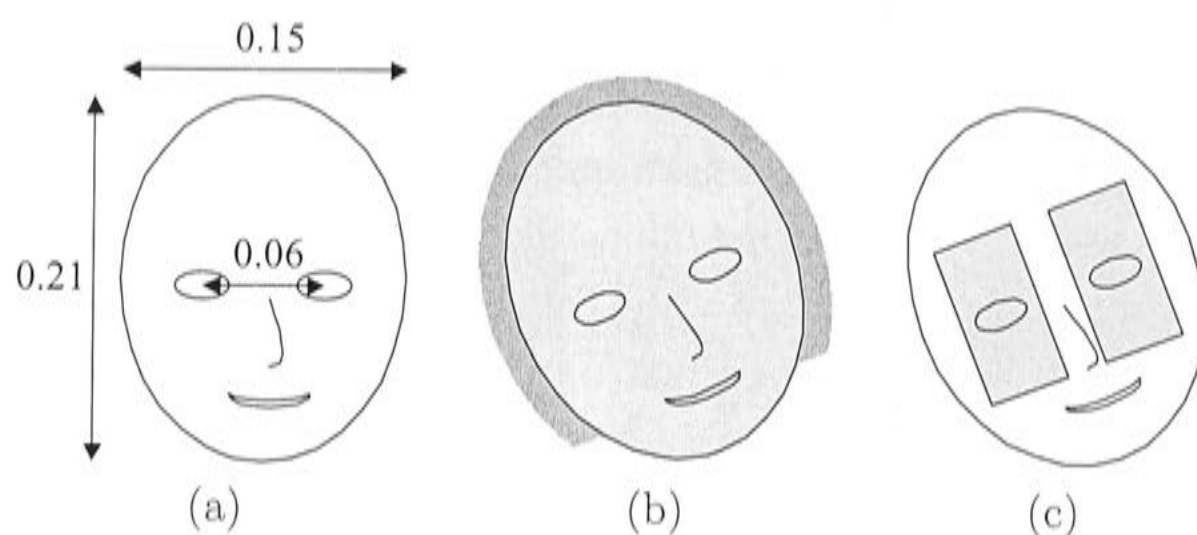


Figure 4.7: Generic head target and associated search regions. (a) Generic head target with dimensions in meters, (b) Elliptical face region (light) and face boundary region (dark), (c) Search regions for integral projection.

indicating the i^{th} active cue's belief in the j^{th} hypothesis (where N is the total number of particles).

The cues were chosen on the grounds of simplicity and efficiency. All cues use the head model dimensions shown in Figure 4.7(a). In the proceeding descriptions the *face region* and *face boundary* refer respectively to the light and dark grey regions in Figure 4.7(b).

Intensity-based Cues

Eye Location Cue: This cue uses integral projection (described in Chapter 2) to search the regions in Figure 4.7(c) of the intensity image for the darkest bands aligned with the lateral axis of the head. A high value is returned if these are close to the hypothesised eye locations.

Radially Symmetric Intensity Cue: The hypothesized depth of the target indicates which radius of facial symmetry should be used. The cue is determined as the value of the appropriate facial symmetry image at the target location.

Radially Symmetric Eye Cue: The generic face model in Figure 4.7 is used to extrapolate the hypothesised eye locations for the current target hypothesis. The hypothesised depth indicates the appropriate scale of radial symmetry that is best suited to eye detection. Using the radial symmetry image at the appropriate scale, the value of the cue is determined as the average value of the radial symmetry image at the two hypothesised eye locations.

Depth-based Cues

We expect the head to stand out as a blob in the depth map. Two cues are applied that together aim to detect head-sized blobs of the appropriate depth.

Head Depth Cue: This cue checks to see if the hypothesised face region is at the appropriate depth. It compares the depths in this face region with the hypothesised depth of the target, returning a high value when these are in agreement.

Head Boundary Depth Cue: This cue measures whether the area surrounding the hypothesised head region is at a different depth from that of the hypothesis. It compares the depths in the face boundary region to the hypothesised target depth, giving a high value when these are different.

Colour-based Cues

Elliptical Skin Region Cue: This cue indicates the likelihood that the hypothesised target region contains a large proportion of skin-like colour. The value it returns is the average skin likelihood of the pixels within the face region.

Skin Detector Cue: This cue detects targets that contain an instance of highly skin-like colour. It returns 0.5 if any of the pixels sampled in the face region had skin likelihood values within the top 10% of values in the current skin likelihood image, and 0 otherwise.

Non-skin Boundary Cue: In general it is expected that the facial region will exhibit a high proportion of pixels with skin-like colouration, whereas the area immediately outside the face will not. This cue aims to capitalise on this scenario by returning a high value if there are few skin colour pixels in the face boundary region. As such the cue will perform poorly if the target is standing in front of a skin-coloured background, but will provide useful information otherwise.

Radially Symmetric Skin Cue: The target is expected to appear in the skin-likelihood image as an approximately round blob of a known radius, and the hypothesised target depth indicates this radius. The value of the cue is given by the value of the skin-based radial symmetry image (of the appropriate radius) at the target location.

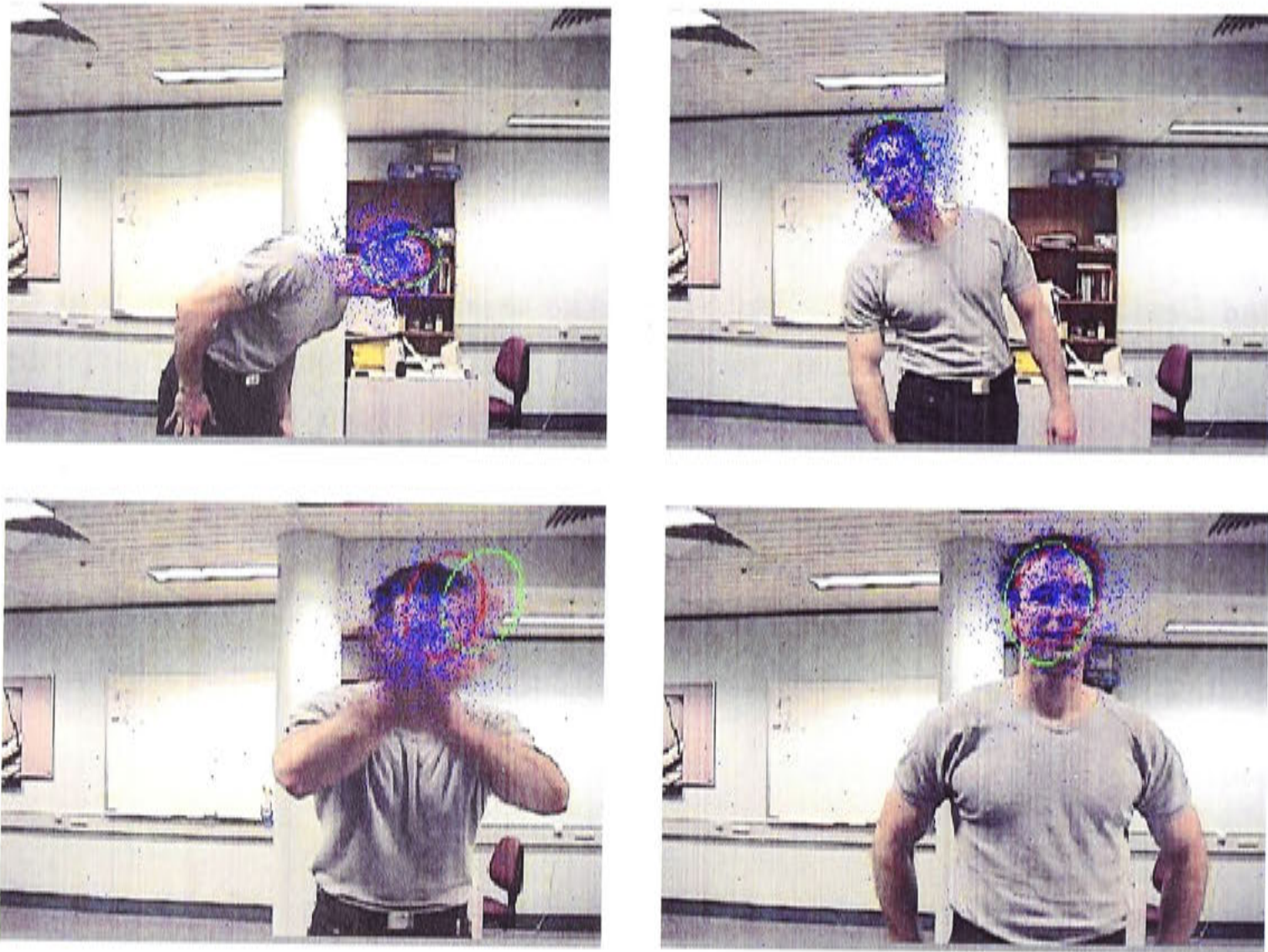


Figure 4.8: Several frames in tracking sequence.

4.3.4 Performance

The performance of the system was demonstrated tracking a human face in two image sequences. Figure 4.8 shows four frames from the first sequence. The blue dots indicate the projected locations of the hypothesised face centres. The hypothesis with the maximum likelihood is indicated as a green ellipse whose size and orientation indicate the hypothesised scale and orientation of the target. Likewise, the expected value calculated across all hypotheses is indicated by a red ellipse.

Figure 4.9 shows a sample frame of the second sequence along with particle distributions. This sequence contains a person moving around a cluttered environment and contains occlusions and lighting variation. Both sequences are included on the enclosed CD-ROM in their entirety.

Cues were dynamically scheduled to run once every 1, 2, 4 or 8 frames according to their calculated utility and computational cost. Figure 4.10 shows the cue utility and processing delay for a specific cue during a tracking sequence. Note that as the cue's utility decreases *relative* to the other cues (i.e., from frames 50 to 80)

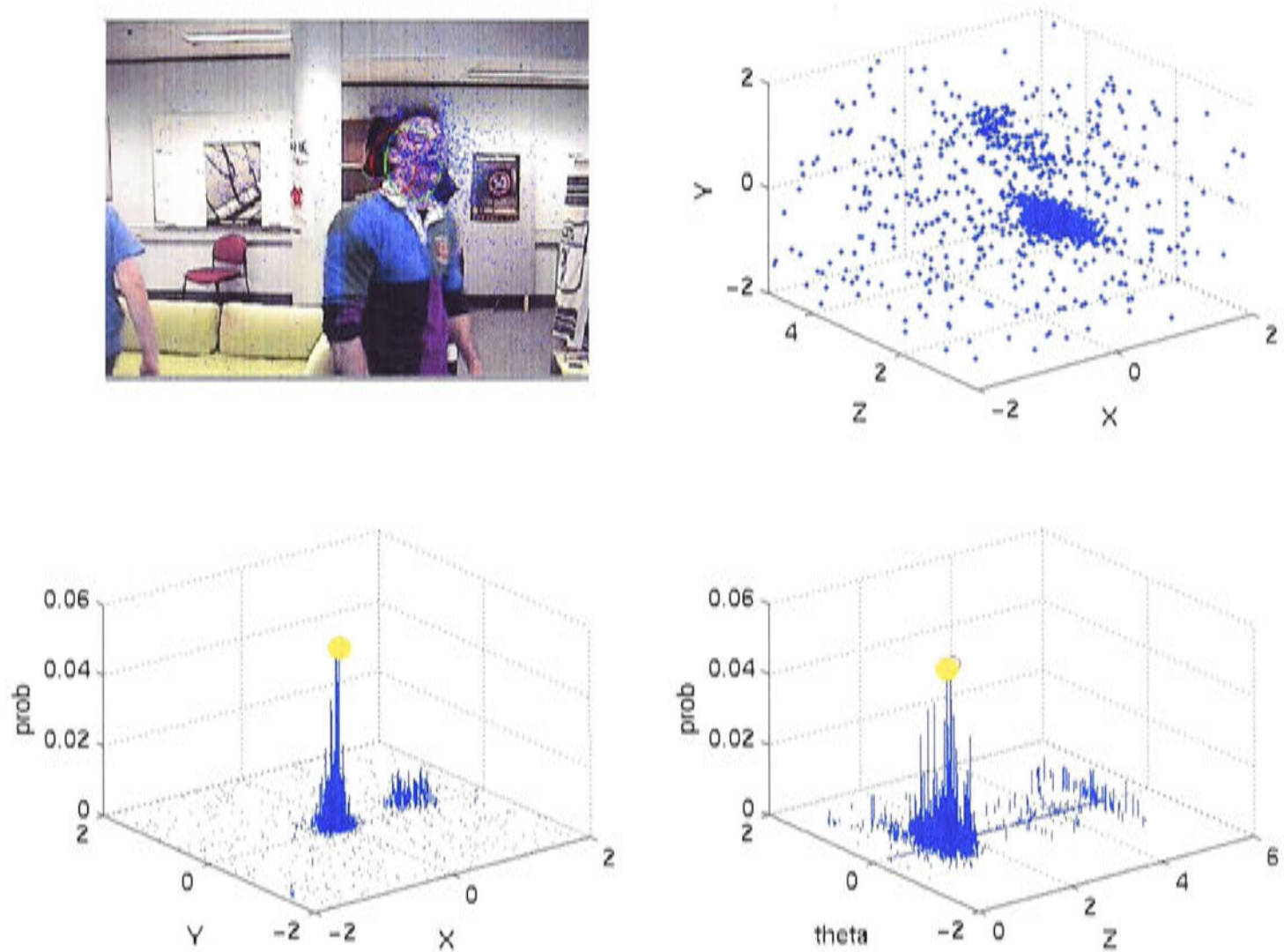


Figure 4.9: Frame in tracking sequence showing (clockwise) particles in image, in 3D space and particle distributions over x , y and z , θ states with the particle with maximum likelihood indicated by a yellow circle.

its processing delay grows as it is allocated less resources.

The simplicity of the cues means no one cue is able to reliably track the head in 3D space, however, by fusing multiple cues the ambiguity in the target location is reduced. Furthermore, by adaptively rescheduling the cues the system was able to enhance the tracking performance possible under a given resource constraint.

Scheduling resources to different cues according to their performance enables a system to aim for the best possible return per unit of computational resource. Without resource scheduling a system is still constrained to use only the computational resources available, yet is ignorant of how to alter computational expenditure to improve performance. Resource scheduling aims to increase the amount of useful information obtained per unit of computation, and — since there is only a finite amount of computation available for each image frame — increase the amount of useful information obtained from each frame.

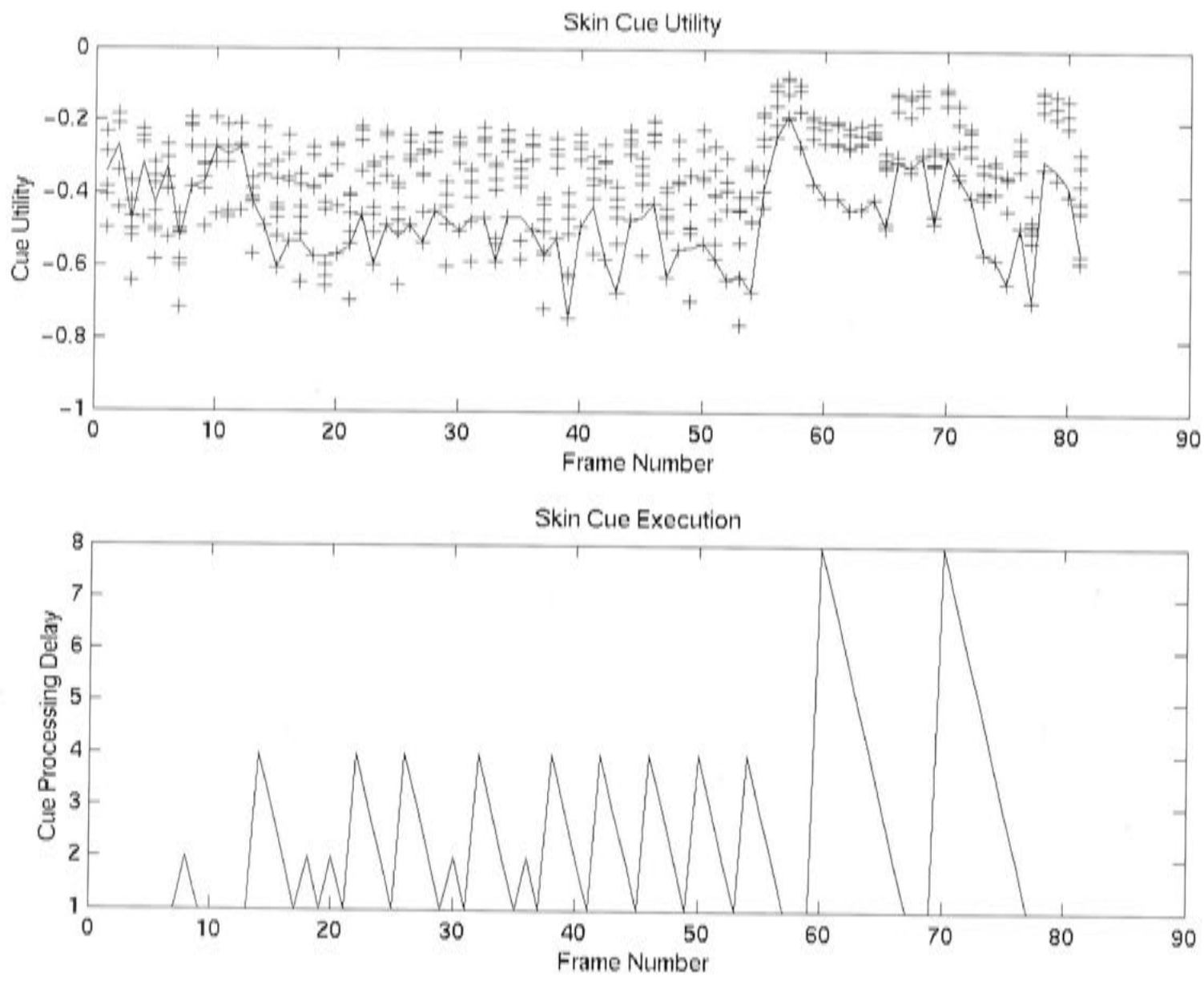


Figure 4.10: Top: Utility for the elliptical skin region cue (solid) and all other cues (+'s) during a portion of a tracking sequence. Bottom: Cue processing delay for the elliptical skin region cue.

4.4 Tracking Multiple Targets

The ability of a vision system to track multiple targets is useful in many circumstances that arise in human computer interaction. It may often be necessary for a system to monitor multiple subjects simultaneously, or track different parts of the same subject, such as the face and hands.

This section² demonstrates how our system can be extended to track multiple targets by running several particle filters in parallel. An experiment is performed showing this multiple particle filter approach successfully tracking a person's face and hands simultaneously.

²The work reported in this section was undertaken whilst visiting Professor Yasuo Kuniyoshi and Dr Gordon Cheng at the Humanoid Interaction Laboratory, AIST, Tsukuba, Japan.

4.4.1 Multiple Particle Filters

Multiple particle filters can be used to track multiple targets by assigning one particle filter per target, and using *inhibitions of returns* to prevent different particle filters from locking onto the same target. In this case inhibition of returns simply amounts to inhibiting the maximum response from each particle filter so that subsequent filters do not give the same response.

Each frame the particle filters are all run sequentially. After each particle filter has run the hypothesis for which it returned the highest probability is blanked out of the evidence for the subsequent filters, so the target it assigned the highest probability becomes invisible to the other filters. The target remains invisible until that filter runs again in the next frame, so it cannot be observed by any of the other filters in the meantime. Note that when the evidence is updated for the new frame the necessary region is blanked out in the evidence for the new frame until the appropriate filter runs again. An example of this process is shown in Figure 4.11 for two particle filters tracking in a one-dimensional state space. The first filter is shown with blue particles, the second with red. After the particle with the maximum response is determined for the first filter the evidence in the vicinity of this particle (indicated by the green shading) is set to zero for the second filter, so any particles from the 2nd filter that fall in this region will return zero probability. Likewise, after the maximum response is determined for the second filter the evidence in the vicinity of this maximum is set to zero for the next iteration of the first filter, etcetera.

4.4.2 Experimental Setup

The purpose of this experimentation was to verify that multiple particle filters could be run in parallel and successfully track different targets. A single colour video camera was used to capture an image sequence at 30 frames per second, the resource allocation was held constant and three simple cues were employed each frame to track the head and hands of a subject. These targets were located in 3D, however with only a monocular system the z depth (in the direction of the optical axis) was not expected to be accurately determined. Unlike the previous experiment orientation was not considered (all cues used were rotationally symmetric about the optical axis).

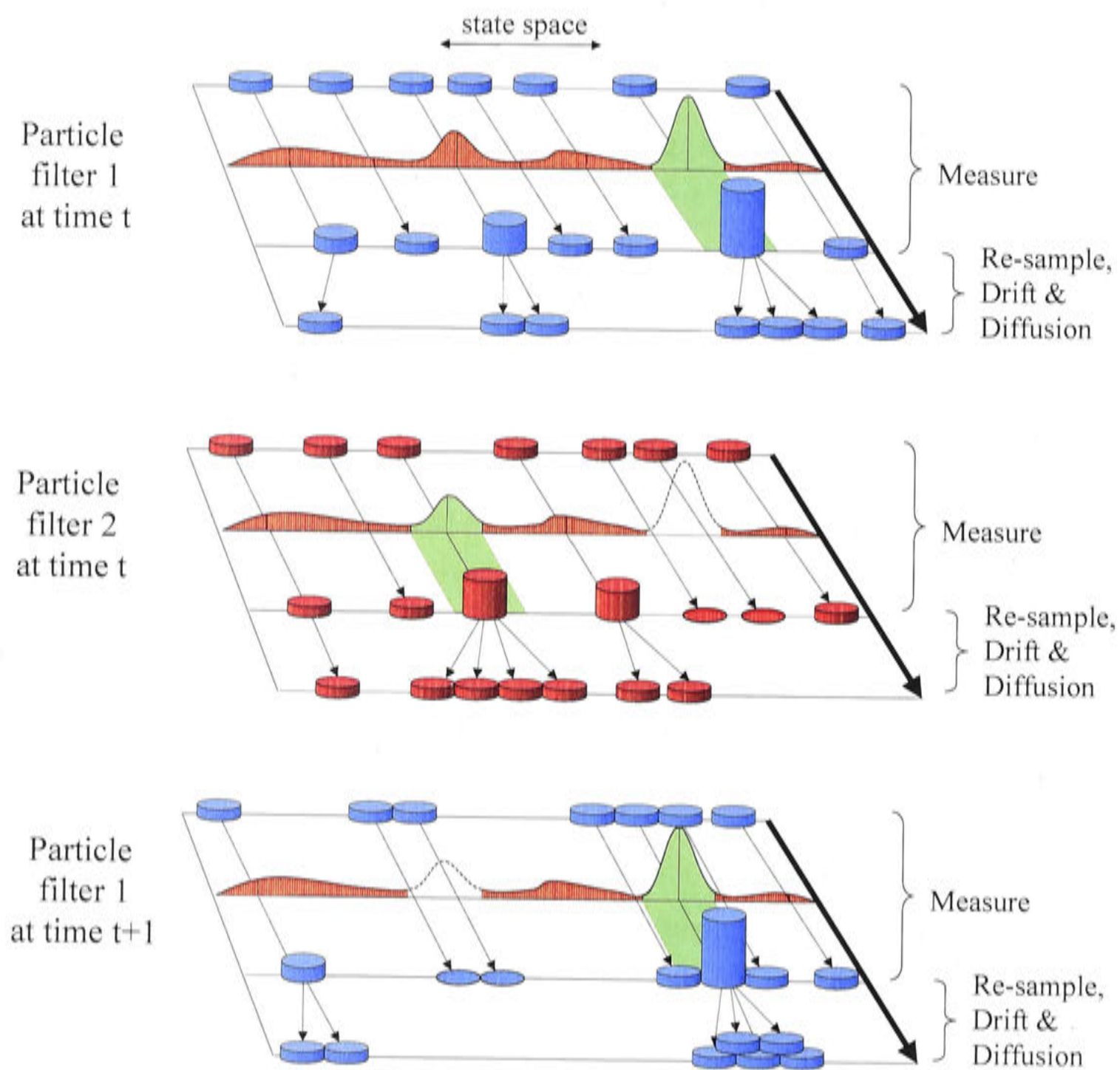


Figure 4.11: Two particle filters tracking separate targets in a one-dimensional state space, using inhibition of returns to prevent both filters converging on the same target. The green shading indicates the inhibited regions around the maximum responses.

Three particle filters were run in parallel to track the three targets of the face and the two hands. Inhibition of returns was used to prevent the different particle filters from locking onto the same target.

Preprocessing

Three preprocessed images are calculated in this experiment:

- Facial symmetry images (computed at three different radii)
- Radial symmetric skin image (computed at three different radii)

- Motion image

The facial symmetry images and radial-symmetric skin images are the same as those used for the Face Localisation experiment detailed in Section 4.3. An additional motion image is introduced that highlights regions of the image that have been changing and are thus likely to contain motion. This is generated using the adaptive background method described in Chapter 2 Section 2.1.4.

Figure 4.12 shows a sample image from a sequences together with the preprocessed images generated. The suitability of the motion image for highlighting regions of movement is clearly demonstrated by the bright regions indicating where the hands have moved as the subject draws the bottle of drink towards his mouth. The skin radial symmetry is also a very effective cue, and whilst there are regions of skin-like colour that do not correspond to hand or face regions (parts of the bookshelf and the keyboard for instance) these are not awarded as high a result since they do not occur as roundish blobs in the image.

Hypothesis Testing

Three simple cues were used to verify the presence of the target:

- Radially Symmetric Intensity Cue,
- Radially Symmetric Skin Cue, and
- Motion Cue

The Radially Symmetric Intensity Cue and the Radially Symmetric Skin Cue are identical to those used in the Face Localisation experiment detailed in Section 4.3.

The motion cue simply returns the value of the motion image at the hypothesised target location.

4.4.3 Results

The results showed that the multiple particle filters effectively lock onto and track separate targets. The simple cues, which were initially designed to locate faces, are also shown to be effective at locating hands.

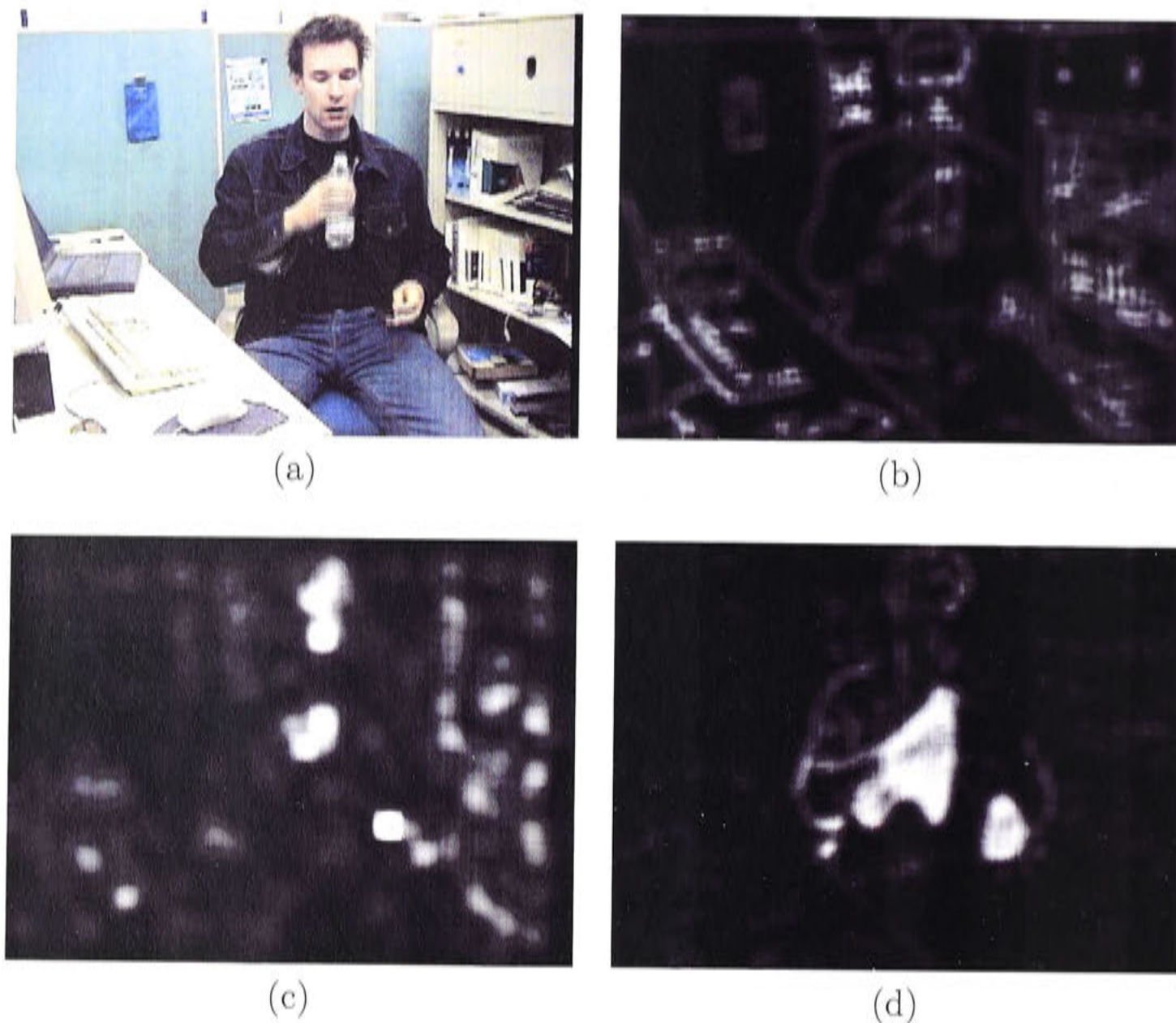
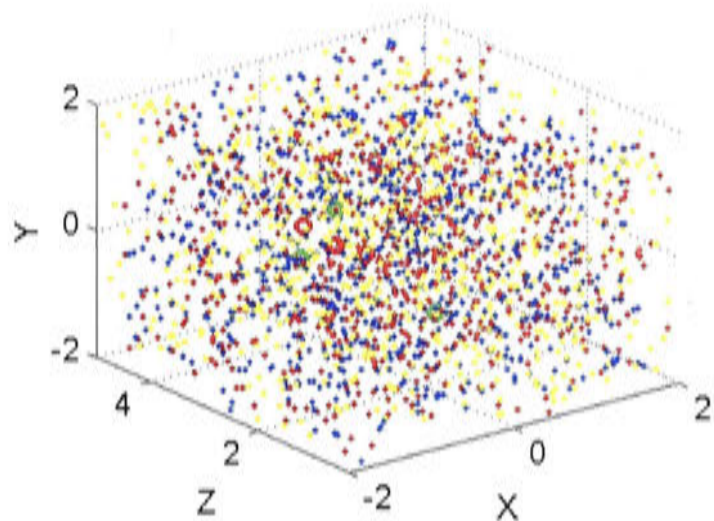
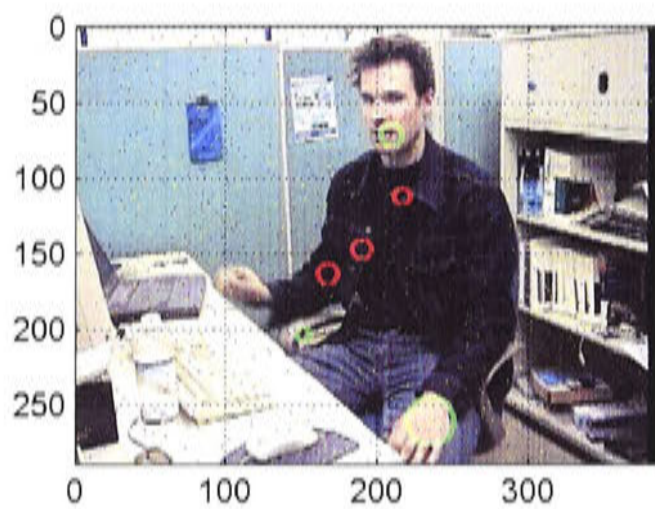


Figure 4.12: Preprocessing results from single camera. (a) Image captured by camera during a sequence, (b) facial symmetry image, (c) skin radial symmetry image, and (d) motion image.

Figures 4.13 to 4.15 show a number of snapshots of the system running over a sequence. This sequence contains a person sitting at a desk in an office environment. He moves around in his chair, reaches and grasps different items, has a drink, and examines a CD case. The complete sequence is contained on the CD-ROM enclosed with this thesis. For each frame of the sequence the input image is shown in the top left with the particle locations of the three filters superimposed on the image. The particles from the three filters are coloured blue, red and yellow so they can be easily differentiated. The hypotheses of each filter with the maximum likelihoods are indicated as green circles, with the radius of the circle indicating the hypothesised scale of the target, and likewise the expected value for each filter, calculated across all hypotheses, is indicated by a red circle. The graph in the top right of each frame shows the distribution of particles across the state space, again green circles are used to indicate the hypotheses with maxi-



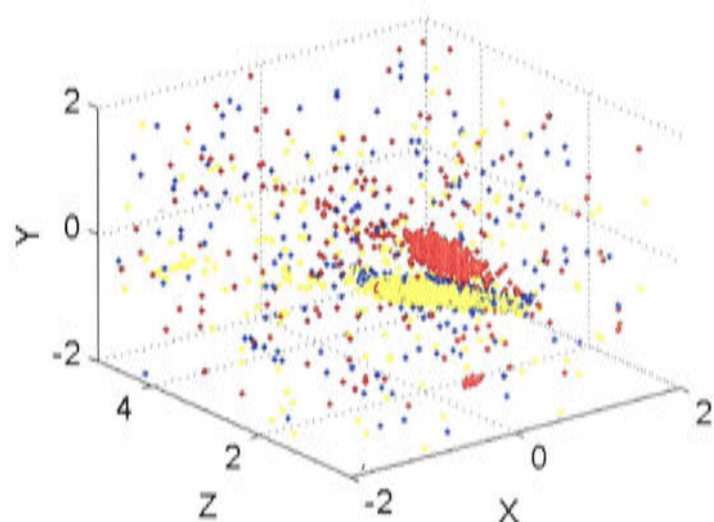
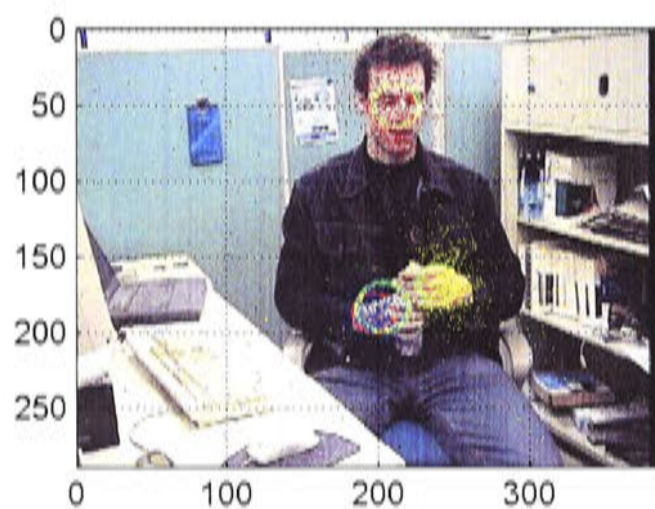
motion

facial symmetry

skin radial symmetry



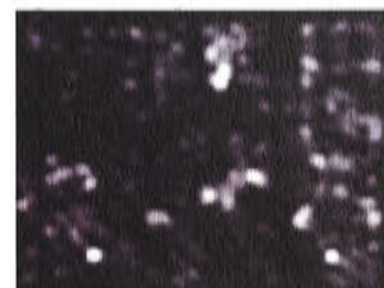
(a) frame 1



motion

facial symmetry

skin radial symmetry



(b) frame 41

Figure 4.13: Some snapshots of the system running over a sequence of approximately 8 seconds duration (234 frames).

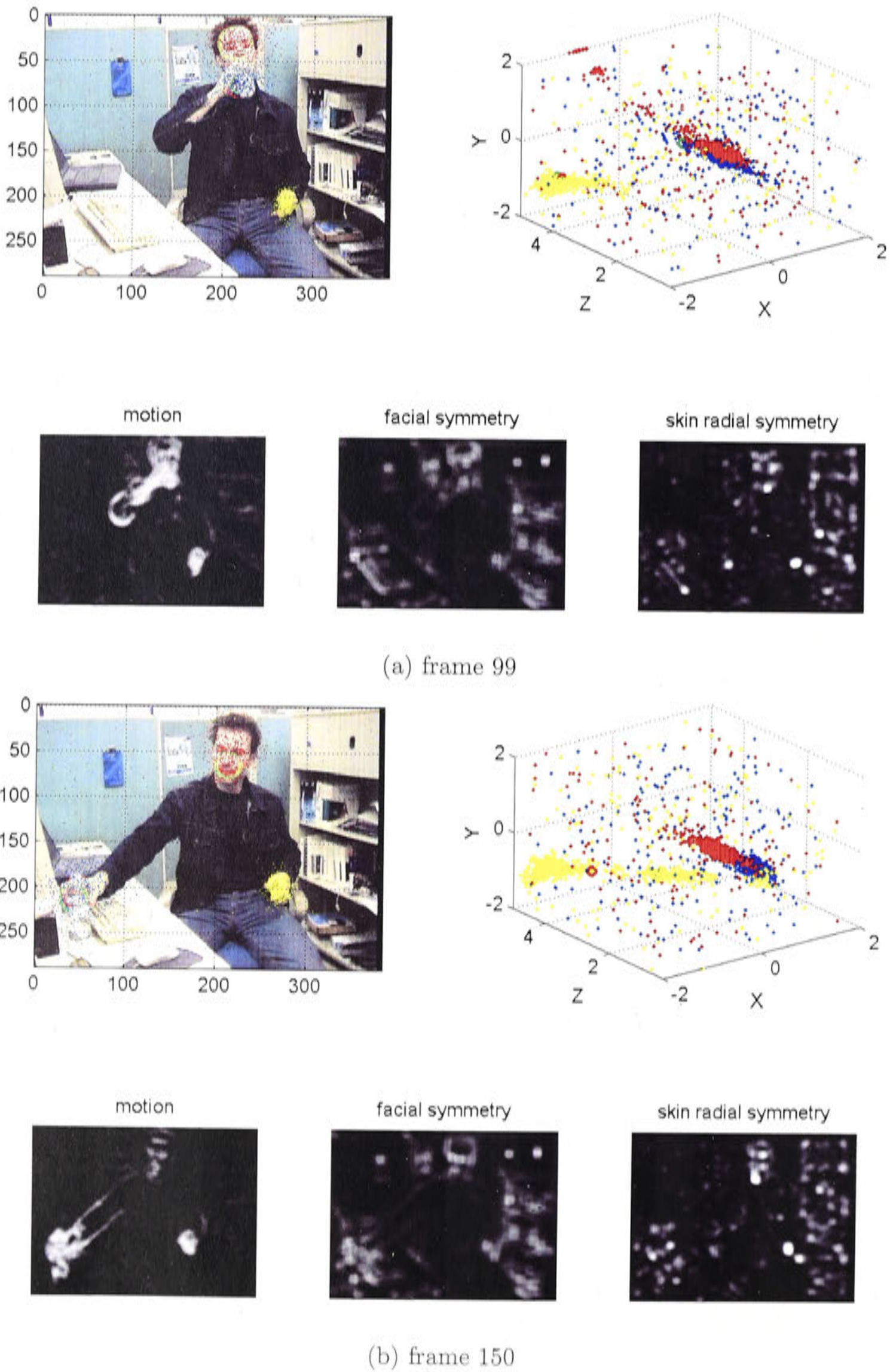
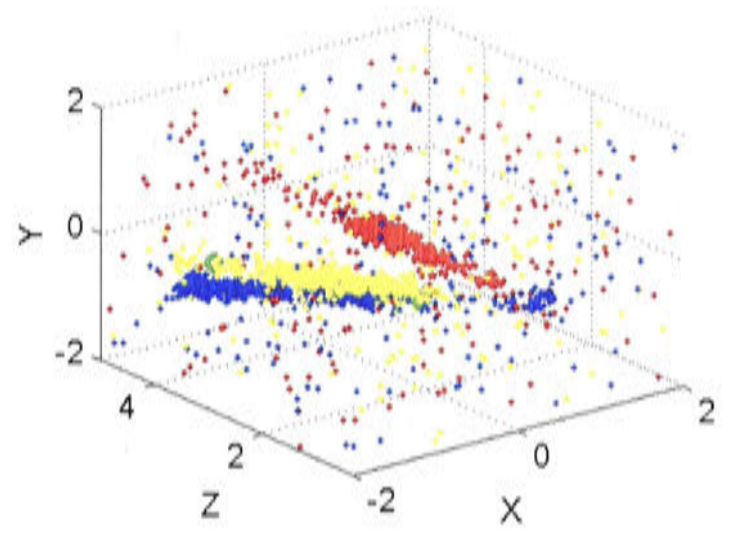
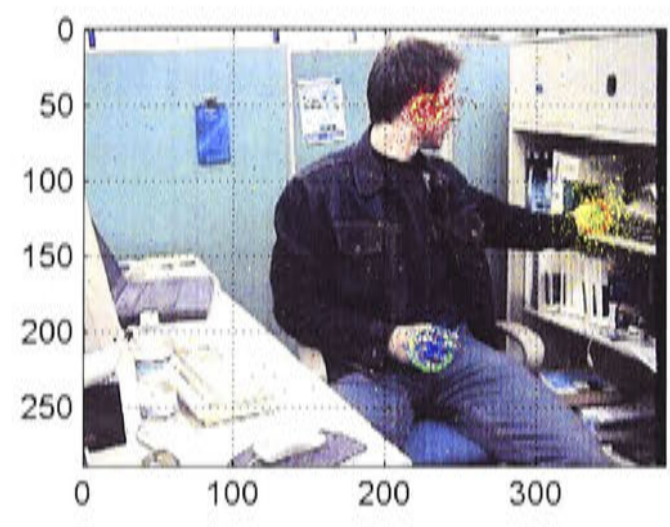


Figure 4.14: Some snapshots of the system running over a sequence of approximately 8 seconds duration (234 frames).



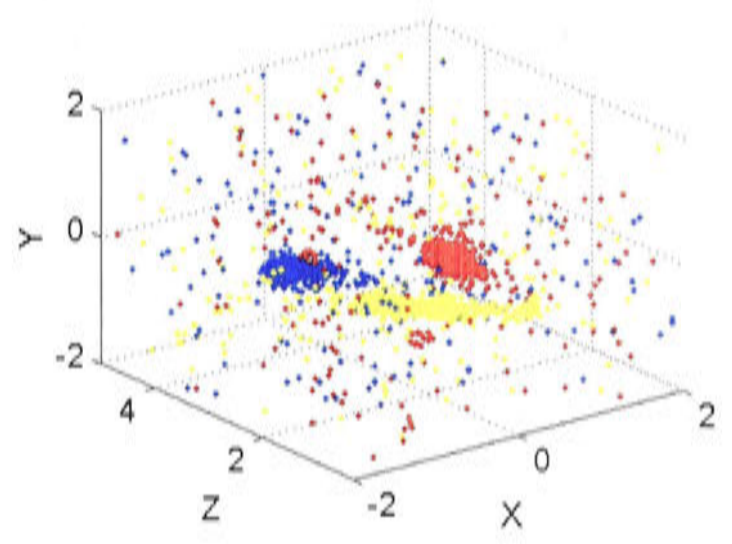
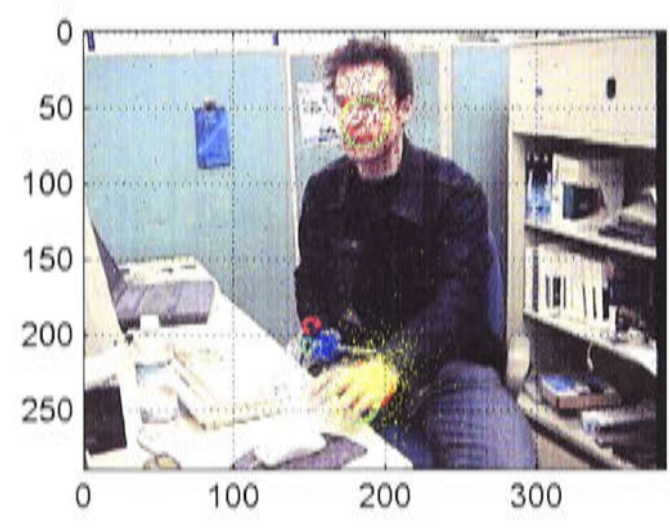
motion

facial symmetry

skin radial symmetry



(a) frame 181



motion

facial symmetry

skin radial symmetry



(b) frame 224

Figure 4.15: Some snapshots of the system running over a sequence of approximately 8 seconds duration (234 frames).

mum likelihood and the expected values indicated with red circles. The three grey-scale images included for each frame show the preprocessed images for that frame.

In frame 1, (a), the particles from all three filters are uniformly randomly distributed across the state space (x, y, z) . Whilst this is the first frame where the particle filters are run it is actually the second frame in the sequence so the motion image is able to be calculated using the previous frame as the decaying average image. By chance one of the filters has correctly located the left hand of the subject and another has located the chin of the face. As the frames proceed the filters rapidly lock onto the hands and face, and proceed to track these targets through a range of motions as the subject reaches around his workspace and manipulates several items.

The filters were only reliably able to locate the target in two of the three state spaced dimensions, with the hypothesised z depths being very ambiguous. This is to be expected from a monocular system with only very crude scale dependance (in the form of the three different radii at which radial symmetry was calculated). However, apart from the ambiguity in the z direction, the system tracked the targets very well. On the few occasions when a target was lost, it was relocated just a few frames later.

One particularly encouraging result was the system's ability to deal with two targets in close proximity without confusing the two. Using inhibition of returns allows for targets to approach each other closely with no danger of the two filters locking onto the same target. The only disadvantage of this is that if one target were to be so close so as to occlude another, then the occluded target could not be detected at all for the duration of the occlusion.

4.5 Summary

This chapter has presented a new approach to target tracking: a vision system that adaptively allocates computational resources over multiple cues to robustly track a target in 3D.

Automatically localising a face and tracking its motion are essential first steps towards enabling a computer to "see" the face. We use a particle filter to maintain multiple hypotheses of the target location and facilitate cues running at differ-

ent rates, whilst Bayesian probability theory provides the framework for sensor fusion. The uniqueness of our system lies in its ability to schedule resources over the suite of available cues. Cues are run frequently or infrequently depending on the usefulness of the information they are providing and the amount of computational resource they require. Keeping short time histories of each hypothesis in the particle filter enables the system to merge information from cues running at different rates.

The system was shown to track a person in 3D space moving in a cluttered environment with variable lighting conditions and occlusions of the target. An additional example was shown demonstrating how the system can be extended to track multiple targets, using multiple particle filters and inhibition of returns to prevent different filters from locking onto the same target.

Chapter 5

Face Registration

FACE registration involves the detection of facial features and the verification of the presence of a face in an image. In the previous chapter we discussed locating and tracking the location of a face in an image sequence. Face registration is the next step towards enabling a computer to see a face. Once face registration is complete the computer has verified whether or not an image region contains a face, and if a face is present, the facial features are detected and ready to be tracked (face tracking is discussed in Chapter 6).

In this chapter we present a case study of an automatic face registration system¹. This system is designed to automatically initialise features for a head tracker. It is required to operate using a single grey-scale video input. This limits the modality of visual cues available, but makes the system suitable to varying hardware and operational environments. The subject is assumed to be within 0.5 and 1m in front of the camera. However, by integrating the face localisation technique presented in Chapter 4 it would be feasible to relax this assumption, allowing the localisation algorithm to identify the approximate location of the face before applying the method describe in this chapter to identify the facial features.

Section 5.1 of this chapter overviews our methodology for automating the feature detection process while drawing comparisons to previous research in the field. In Section 5.3 we give a detailed description of our algorithm for detecting facial features, the performance of the algorithm is demonstrated in Section 5.4, and Section 5.5 closes with a summary of the key points.

¹This chapter reports the findings of commercial research undertaken in a consulting capacity for Seeing Machines. The research is included in this thesis with the consent of Seeing Machines on the condition that this chapter be embargoed for twelve months after submission of the thesis.

5.1 Automating the Detection of Features

Automatic feature detection for tracking is essential for face tracking systems to be able to cope with new users without prior knowledge of the user's appearance. The key problem with facial feature detection to date is robustness. A system is required that is robust to the wide variety of human appearances and the varying lighting conditions of operational environments, and that operates in realtime.

Many recent face detection systems rely heavily on skin colour to locate the face (Sobottka and Pitas, 1996b; Cai and Goshtasby, 1999; Kim and Kim, 2000). They are subsequently highly sensitive to lighting conditions, and unable to operate at night using monochrome images from an infrared camera. We have developed a system that requires only monocular monochrome images, giving both maximum robustness to lighting changes and the versatility to function with a wide range of image-capturing devices.

Our system requires the user to blink to initiate the feature detection process. Blink detection has been shown to be a useful cue for locating the eyes in video sequences (Crowley and Berard, 1997; Bala *et al.*, 1997). Consecutive frames are differenced to determine regions of motion, and two blink-like motion regions are located and labelled. Previous implementations have simplified the problem by using skin colour detection to identify the face region before looking for blinks (stereo depth information has also been used (Bala *et al.*, 1997)), however, the monocular grey scale input to our system restricts us from using these additional modalities. Previous systems have not considered whether the detected eyes are open or closed — this is important for our system since we wish to detect appropriate features for tracking a subject whose eyes will be open the majority of the time.

The radial symmetry detection algorithm presented in Chapter 3 provides an alternate means to obtain estimates of eye location independently of motion. However, given that motion information is available, blink detection gives a robust means of utilising this additional sensing modality and ensuring with greater certainty that the eyes are correctly located. The radial symmetry operator is subsequently used to generate a mask of the regions containing facial features.

Once a potential face region is found, and the facial feature mask constructed, our system uses a novel local comparison operator to highlight features of interest. This operator employs a similar principle to the rank transform introduced by

Zabih and Woodfill (1994) for the purpose of pre-processing images before calculating stereo correspondence. As discussed in Chapter 2 Section 2.1.3, Zabih and Woodfill's rank transform was calculated for a pixel p by counting the number of pixels in a local region centred on p whose intensities were darker than the intensity at p . The result was an increase in local texture in featureless areas of the image. The extension of this principle presented in this chapter has quite a different effect, and highlights facial features based on their comparative brightness when compared to other parts of the face.

Integral projection plays a key role in the localisation of potential feature candidates in our system. Integral projection and variations thereof have been used to detect facial features in a number of applications (Yang *et al.*, 1998a; Katahara and Aoki, 1999; Chuang *et al.*, 2000). The main problems are segmenting the region of interest from the image to avoid background interference, and ensuring that the desired features stand out to the exclusion of everything else. Our system addresses these problems by extracting the face region based on blink locations, and enhancing the features in the candidate face region, before performing integral projection.

5.2 Target Specification

We are interested in detecting faces and facial features. In Chapter 2 we considered various properties of the human face with particular emphasis on characteristics that could be detected with computer vision and used to locate faces. In this section we develop a model describing the relative locations of facial features in a frontal view of the face. The model is based on the average face and facial dimensions presented in Chapter 2. This model will be applied later in the chapter, together with some of the feature detection methods discussed in Chapter 2, to detect facial features in a monocular grey scale image sequence.

Figure 5.1 shows an image of the average face (refer to Chapter 2) together with the mean locations for facial features and their standard deviations (note these results were averaged over the male and female populations reported in Table 2.1).

Using these dimensions we can develop a set of rules describing the appropriate location of facial features in a front-on view of a subject. The measurements available are measured from the top of the head. However, the deviation of feature

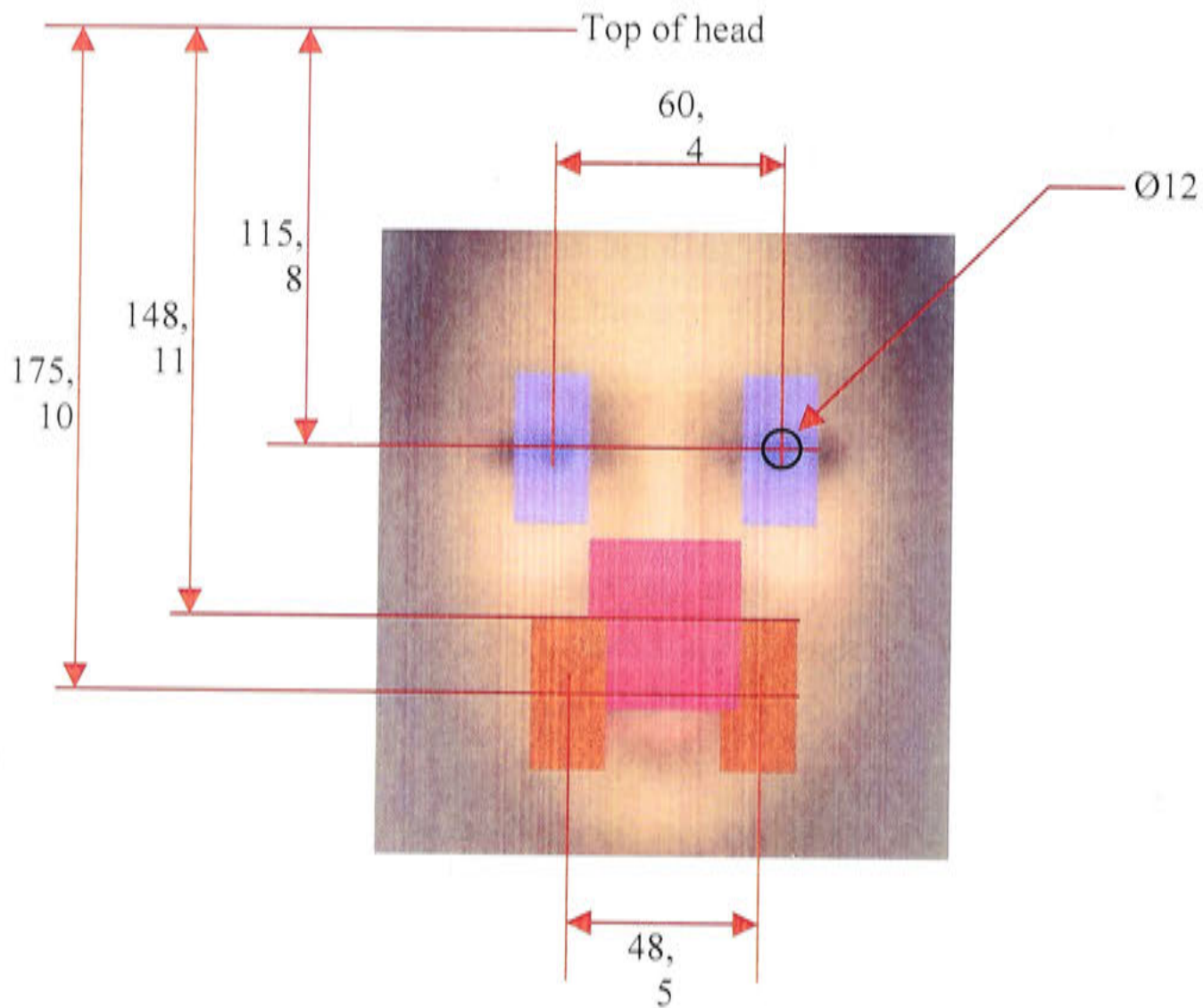


Figure 5.1: Average face and facial dimensions in millimeters, the standard deviation is shown beneath each measurement. The shaded regions show feature locations within two standard deviations of the mean.

locations from their mean values will be correlated between different features, i.e., for larger and smaller headed people the distances of facial features from the top of the head will increase and decrease respectively. Whilst this will by no means be an exact correlation it is highly unlikely that if the mouth is two standard deviations above the mean (at the top of the orange region in Figures 5.1 and 5.2) that the eye will be lower than the mean eye location. Thus, if we assume the eyes are located on the mean eye line shown in Figure 5.1 then the distance from the eyes to the mouth should be at least $0.67d$, where d is the interpupillary distance. Using the same reasoning, and the knowledge that the nose must lie between the eyes and the mouth, we insist that the nose must appear no higher than one third of the way from the eyes to the mouth. We also require that the nose not be within one sixth of this distance from the mouth, as it is not possible for the nose to appear this close to the mouth. The likely width of the mouth can also be estimated from the interpupillary distance as approximately $0.8d$.

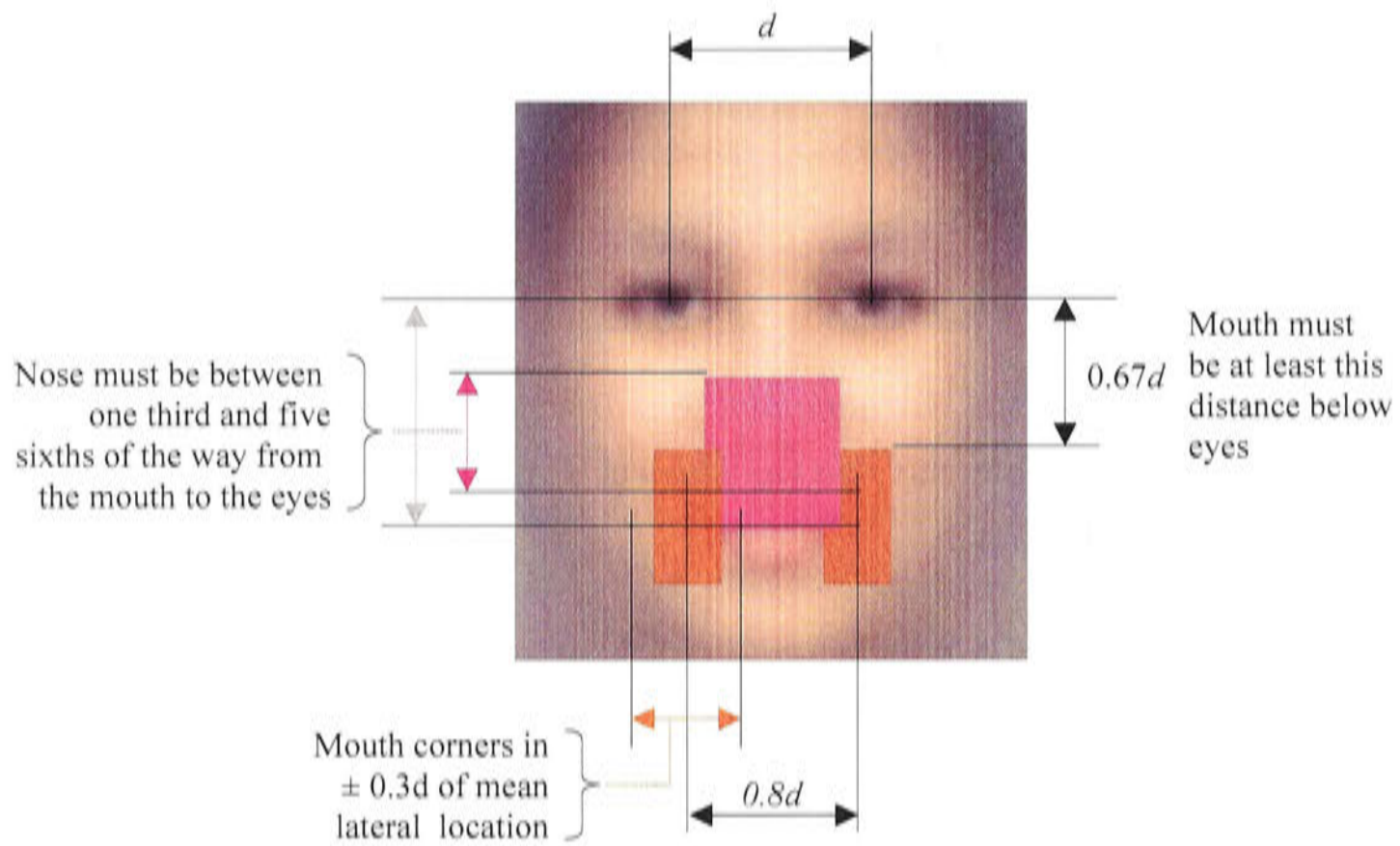


Figure 5.2: Average face showing placement of the mouth and nose. The shaded regions show feature locations within two standard deviations of the mean.

However, mouth width can vary significantly with different facial expressions so the mouth corners are only assumed to lie within $\pm 0.3d$ (approximately 3 standard deviations) of these locations. These requirements are illustrated in Figure 5.2.

The purpose of our system is to automatically locate the face and facial features for a face tracking application. We can assume the subject is seated in front of the camera with the camera looking slightly upwards at his or her face. Since our algorithm is designed to work on a static monocular system there is no means of directly measuring the depth and establishing the scale of objects in the image. For this reason we provide the algorithm with a scaling factor indicating the approximate radius of the subject's iris in the images. This radius value, referred to as r , allows the system to search for faces of the correct size in the input images. It is a simple matter to adjust r if the set up is changed, for example, by using different cameras, or requiring the subject to sit a different distance away from the camera.

The systems can be extended to use stereo cameras thereby eliminating the need to provide the scaling factor r . This could be achieved by using stereo depth



Figure 5.3: Structure of face-finding algorithm.

information to determine the distance of the subject from the camera, which in turn enables us to determine the size the subject's iris will appear in the image, hence providing the constant r .

5.3 Description of the System

This section describes our algorithm for locating the facial features in a sequence of video images. Figure 5.3 shows the structure of the algorithm. Pairs of points exhibiting blink-like motion are detected, and from these points a candidate face region is determined and extracted from the image sequence. The features within this region are enhanced and classified as potential facial features. The topology of the resulting features is then examined to determine if a set of valid face features has been found. If at any point during this process it becomes evident that the features do not represent a face, then the candidate face region is discarded and the system returns to looking for regions of blink-like motion in subsequent frames. The process is repeated until a suitable set of facial features is detected.

The algorithm requires the scaling parameter r (the expected iris radius in pixels) to define the scale at which faces are detected. It can cope with some variation in scale so only a rough estimate of r is required, and it is not necessary to update r for different subjects. Updating r is only necessary when images of a different resolution are used, or if subjects are to be detected at significantly different ranges away from the camera.

5.3.1 Detecting Blink-like Motion

The average adult blinks ten to fifteen times a minute, or once every four to six seconds (Stern, 2002). Blinking is a rapid and distinctive motion occurring at both eye locations simultaneously. When eyes blink the transition from open to closed eye is typically very fast, and in an image sequence captured at 30Hz

(NTSC video frame rate) generally occurs in the time between one frame and the next. Therefore, by examining the change in consecutive frames in an image sequence, potential blink locations can be identified by looking for eye-sized region pairs where the image has changed since the last frame.

Detecting blink locations is a simple and robust method of eye detection, and is a suitable cue to use to initiate face detection. Apart from the natural regularity of blinks, it is not unreasonable to request a subject to blink to commence the face detection process, or more subtly, blinks can be induced by manipulating a display in front of the subject. The only minor issue with requesting a subject to blink is that when subjects blink intentionally the transition from open to closed eyes tends to be slower and more deliberate than for involuntary blinks, thus voluntary blinks are less robustly detected by image differencing consecutive frames. For the purpose of the algorithm presented here the subject will be assumed to be blinking involuntarily and hence making the transition from open to closed eyes in a single frame.

The process for detecting blink-like motion is outlined in Figure 5.4. For each frame a binary motion image \mathbf{M}_t is constructed identifying regions of significant motion. The motion image is calculated by first taking the absolute difference between the current and previous frames, low-pass filtering this via convolution with a Gaussian, and then applying a threshold. A record of this motion is stored in a motion history image \mathbf{H}_t , each element of which indicates how long ago motion was seen at that point of the image.

Once we have obtained the binary motion image \mathbf{M}_t we can identify separate regions of motion using a sequential scanning algorithm (Horn, 1986, e.g.) to uniquely label all 8-connected² regions of motion. The height, width and centre of mass of each of these regions is then determined.

We prefer to locate open eyes, so we try and identify only the initiation of a blink, and then look for the eyes in the frame 0.1 seconds before the motion was detected. To avoid detecting the final (re-opening) motion of a blink, we require there to have been no movement at the blink points between 0.1 and 0.3 seconds prior to the motion being detected (since blink durations are typically less than 0.3 seconds).

The motion history image \mathbf{H}_{t-3} is used to verify this. This image is shown on

²A pixel is 8-connected to all 8 pixels in its immediate neighbourhood.

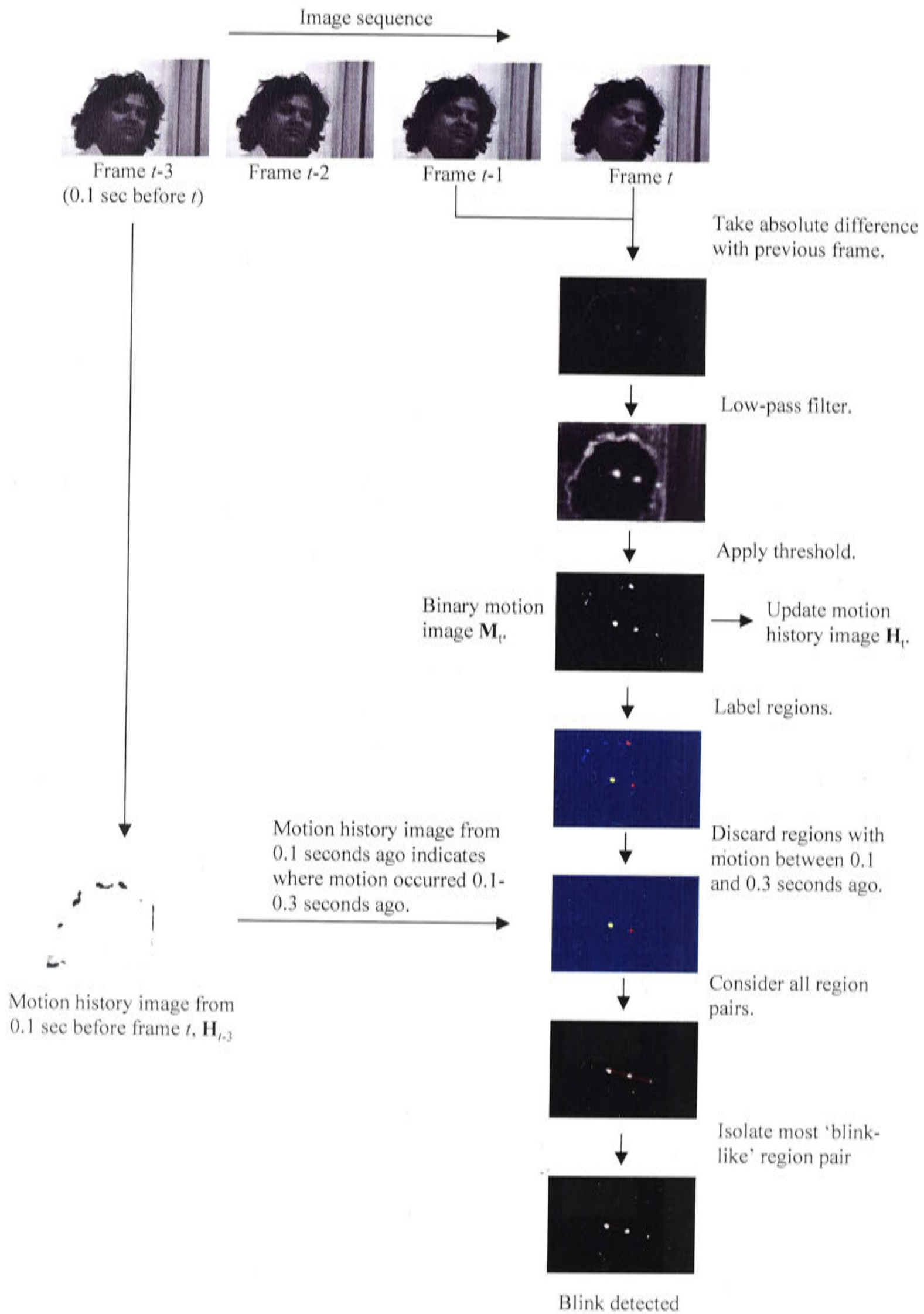


Figure 5.4: Detection of blink-like motion.

the left hand side of Figure 5.4, and the value at each pixel indicates how many frames have elapsed since an \mathbf{M}_t image registered motion at that pixel location. We are only interested in motion between 0.1 and 0.3 seconds ago, so the range of values of the motion history image is 0 to 6 with 6 indicating that no motion has been observed for 6 frames (0.2 seconds). The motion history image in the figure is predominantly white, indicating that most locations have not registered motion for six or more frames. We check the value of this motion history image \mathbf{H}_{t-3} at locations corresponding to the centres of mass of each of the regions of motion in the current frame, and discard those regions that do not have a motion history value of 6, i.e., we discard those that have exhibited motion in the last 0.1 to 0.3 seconds.

We then consider the sizes of the remaining regions and check that these are not too large or small to correspond to blink regions. We use fairly generous criterion with the aim of minimizing the number of false rejections at the expense of more false positives. We require the width of the blink region to lie within $0.5r$ and $6r$, the height to be less than $6r$, and the width to be greater than the height (here r is the radius of the iris).

A set of region pairs is constructed containing every pair of regions that is an appropriate distance apart to be an eye pair, and whose centres of gravity are joined by a line less than 30 degrees to the horizontal (this is more than sufficient to accommodate for natural inclination of the head from the upright position). The pair whose regions have the most similar heights, widths and areas is selected as exhibiting the most “blink-like” motion, and the centers of the two areas of blink-like motion are called *blink points*.

If a valid pair of blink points is found, their locations are used to define the face region that is extracted (from the image frame 0.1 seconds before the motion was detected) in the next stage of the algorithm.

5.3.2 Extraction of Face Candidate Region

The possible eye locations estimated by the blink detection step described above are used to specify and extract a potential face region for further processing. This region is extracted from the image frame 0.1 seconds before the blink-like motion was detected. The reason we look at the frame 0.1 seconds prior to the blink is that we wish to detect open eyes, and the eyes will be open 0.1 seconds prior to

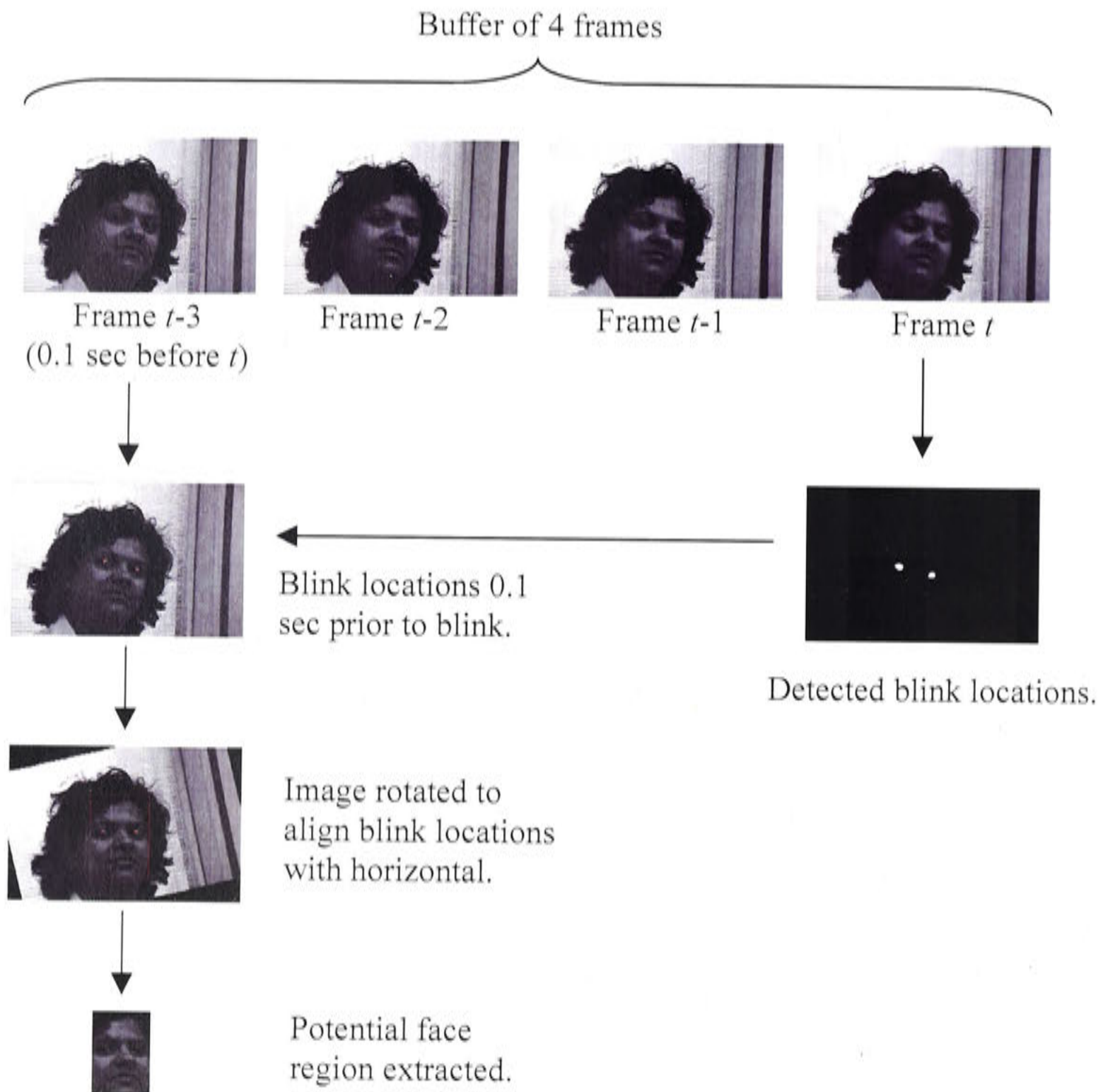


Figure 5.5: Process for extracting potential face region from image buffer.

the initiation of a blink.

The process of extracting the potential face region is illustrated in Figure 5.5. A four-frame buffer is maintained in order to extract the face region 0.1 seconds (four frames) prior to the detection of the blink. Once the blink points have been located in this earlier frame, it is rotated to align the blink points with the horizontal (bi-linear interpolation is used to calculate the pixel intensities of the rotated image). This normalises the orientation of the face candidate to an upright position. A rectangular face region is then defined as shown in Figure 5.6. The dimensions of this region are chosen so it will include all the facial features, and its size is scaled by the interpupillary distance between the centres of the

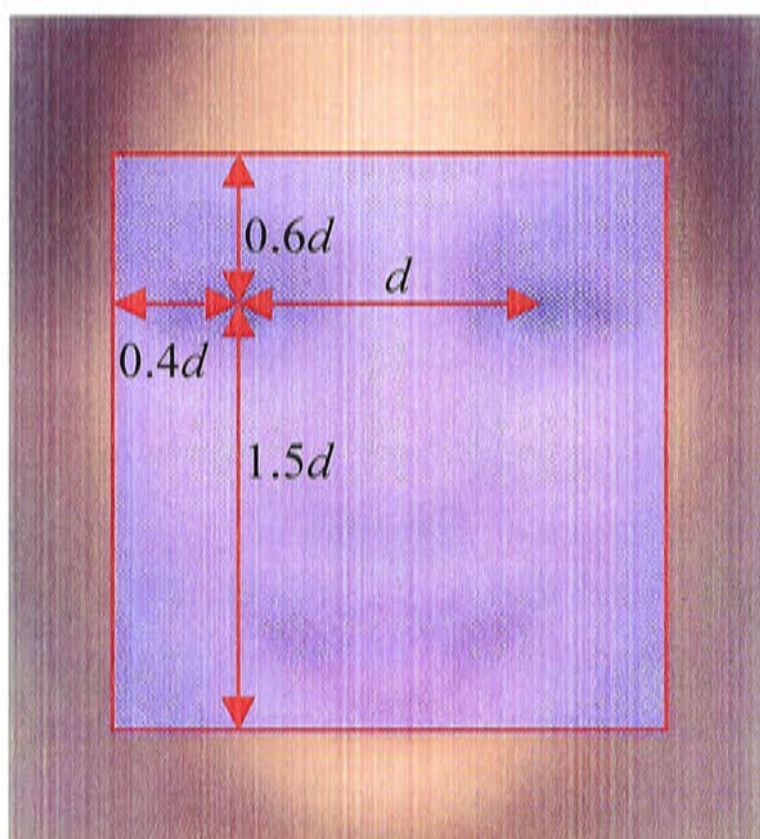


Figure 5.6: Face region defined in terms of d , the interpupillary distance between the centres of the blink points. Note how this region encompasses all facial features of the average face.

blink points. This region is then cropped from the rotated image and becomes the *face candidate region* that is passed to the next phase of the algorithm for feature enhancement.

5.3.3 Enhancement of Features

It is now necessary to enhance the features so they stand out more distinctly from the background to make them more easily detected via integral projection. The process for enhancing features is outlined in Figure 5.7. The face region in Figure 5.8 will be used as an example to illustrate the different steps of the procedure.

Firstly a face mask is constructed that contains all the facial features and as little of the rest of the face as possible. This is done using local radial symmetry. Radial symmetry peaks in the vicinity of facial features, and has been used for facial feature detection in several applications (Reisfeld and Yeshurun, 1998; Lin and Lin, 1996; Sun *et al.*, 1998). We estimate local radial symmetry at each point in the face region based on gradient orientation, using the fast radial symmetry transform, defined in Chapter 3. The result is shown in Figure 5.8. Orientation-

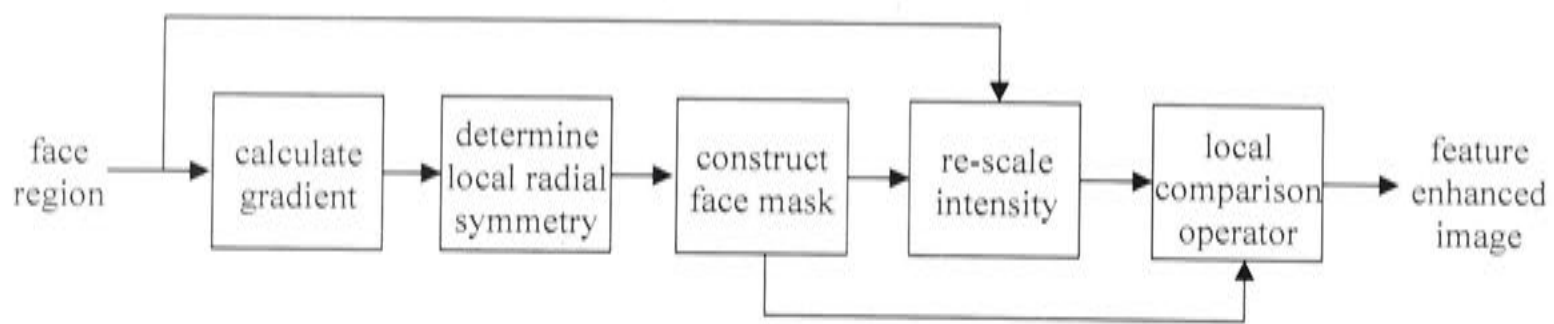


Figure 5.7: Process for enhancing features in face region.

based radial symmetry is used due to its greater robustness to varying lighting conditions. Whilst this transform can accurately pinpoint features in many instances we do not wish to rely on this one mechanism alone for locating features. Therefore we blur the result with a horizontally opposed rectangular Gaussian mask³ in order to spread the peaks of radial symmetry over the whole of the (predominantly horizontal) features, and then threshold to form a binary mask. The blurred orientation symmetry will not have strong distinctive maxima, and experimentation on numerous images has shown that applying a threshold of 30% of the maximum value will admit all regions that have exhibited sufficient radial symmetry to be features. The resulting mask will cover the eyes, nose and mouth corners but may miss the centre of the mouth due to its lack of local radial symmetry. In order to ensure the whole mouth is included we augment the mask with an additional $0.5d \times d$ rectangle centred d below the eyes, where d is the interpupillary distance between the blink points. This gives the final face mask shown in Figure 5.8.

We then re-scale the intensity of the face region in order to maximise the dynamic range of features within the area defined by the face mask, and attenuate the relative strength of extreme intensity values outside this region. We determine the maximum and minimum intensity values within the area of the face region that passes through the face mask, and use these as upper and lower bounds to re-scale all intensities in the face region to the interval $[0,1]$. Any intensities below or above these bounds are truncated to 0 and 1 respectively (since the bounds are the maximum and minimum intensities in the face mask region this attenuation will only occur at points outside the face mask region).

Features within the intensity re-scaled image are then enhanced by the application of a novel *local comparison operator*. The purpose of the local comparison

³The Gaussian is $r + 1$ high, $3r + 1$ wide with standard deviations of $3r$ and r in the x and y directions respectively, where r is the radius of the iris.

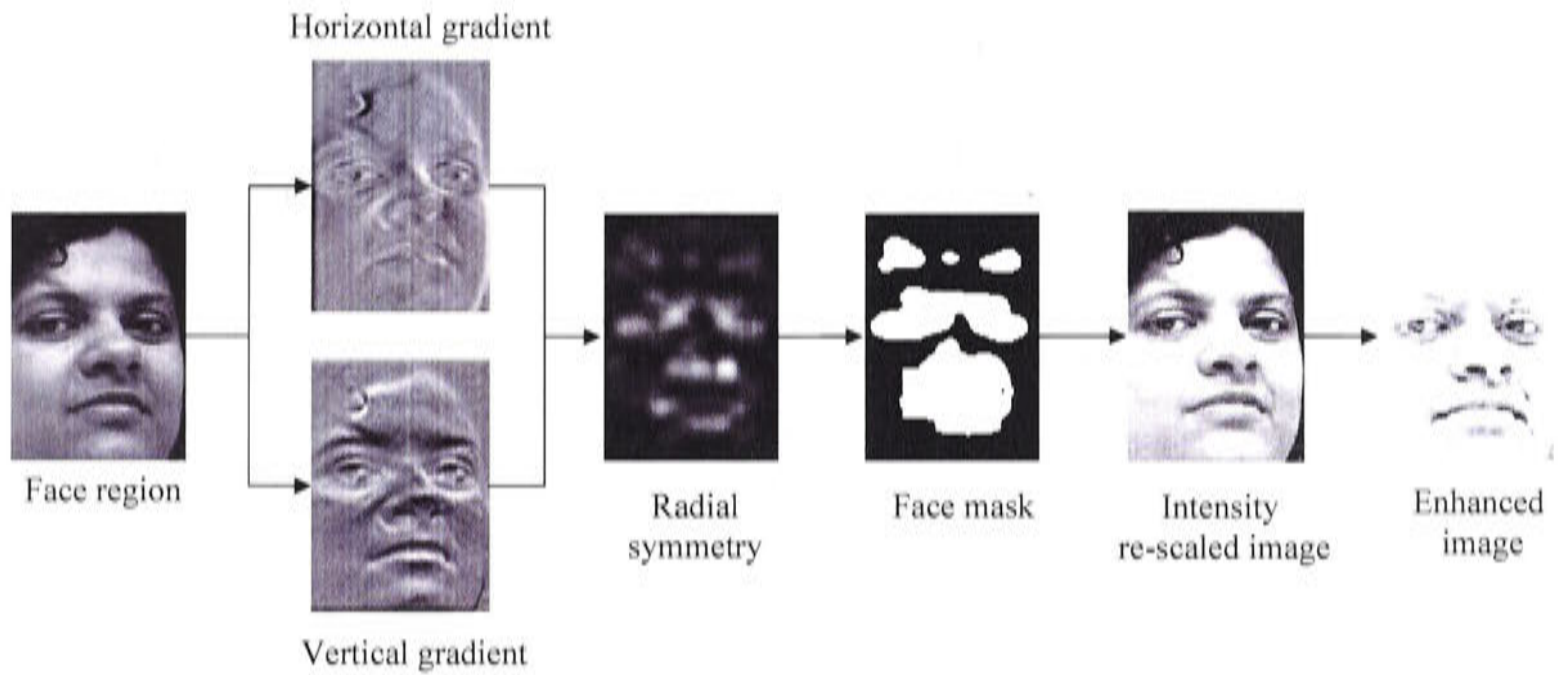


Figure 5.8: Images associated with the enhancement process.

operator is to highlight points that are darker than specified neighbouring regions. The value of the operator is calculated at each pixel \mathbf{p} by comparing the intensity of that pixel with the intensities of each member of a set of pixels $S(\mathbf{p})$ whose locations are defined relative to \mathbf{p} . The operator quantifies the proportion of these pixels whose intensities are greater than $\mathbf{I}(\mathbf{p}) + k$, where $\mathbf{I}(\mathbf{p})$ is the intensity at location \mathbf{p} and k is a constant called the *minimum difference threshold*. That is, for a specified set of pixels $S(\mathbf{p})$ the local comparison operator returns the proportion of pixels in $S(\mathbf{p})$ that are darker than the intensity at \mathbf{p} by an amount k or more.

We formally define the local comparison operator as

$$\mathbf{L}_{S,k}(\mathbf{p}) = \frac{\|\{\mathbf{q} : (\mathbf{q} \in S(\mathbf{p})) \cap (\mathbf{I}(\mathbf{q}) > \mathbf{I}(\mathbf{p}) + k)\}\|}{\|S(\mathbf{p})\|}$$

Here $\|\dots\|$ indicates cardinality, that is, the number of elements in a set.

To enhance possible facial features in the face region we use the sets of points $S_i(p)$ illustrated in Figure 5.9.

The motivation behind the choice of this set of regions is to highlight points that are:

- darker than most of the pixels in the neighbouring region S_4 below,
- darker than most of the pixels in a region S_1 located above and slightly further away, and

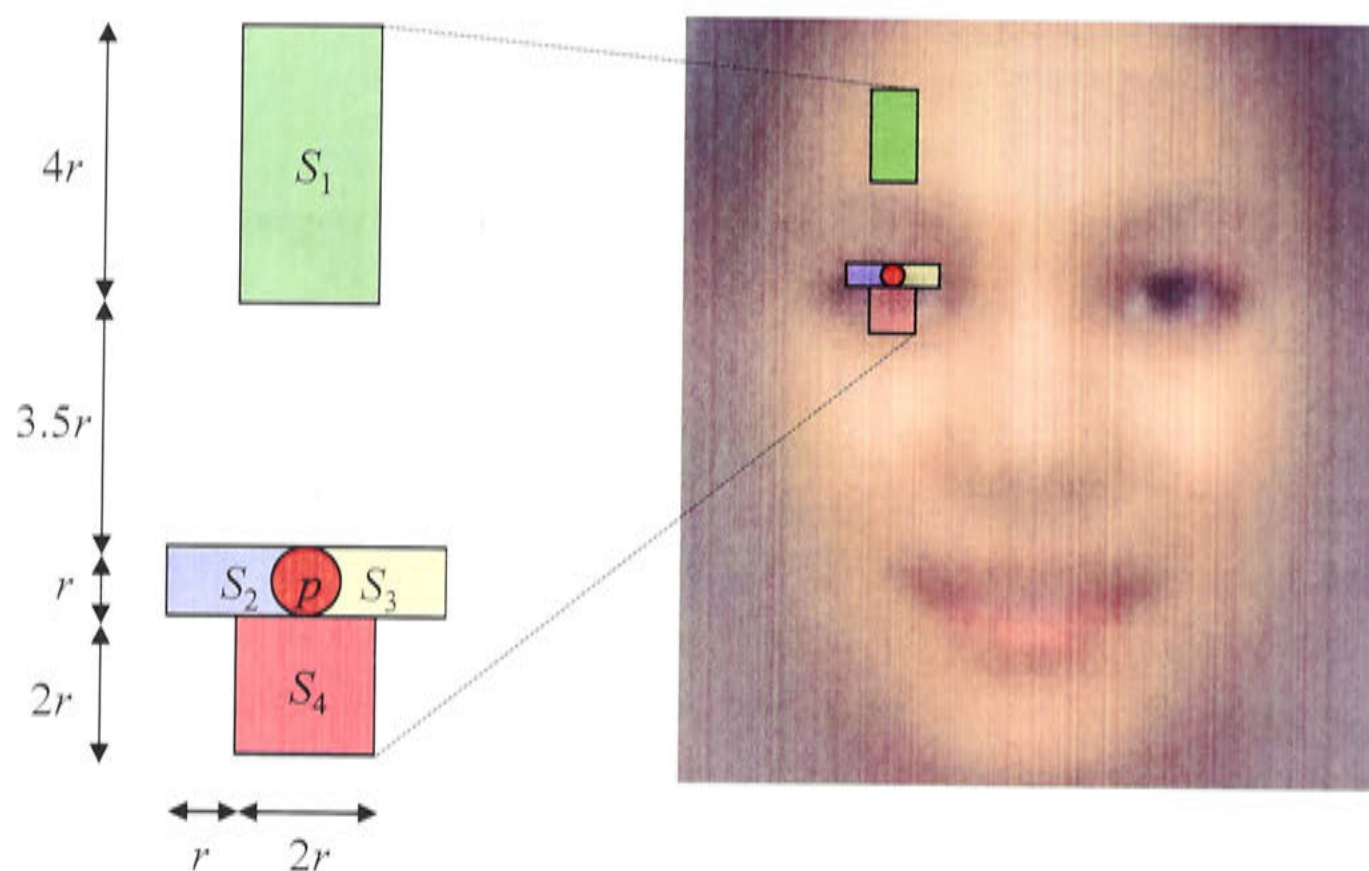


Figure 5.9: Regions S_i used by \mathbf{L}_i for enhancing facial features, r is the iris radius in pixels.

- significantly darker than most of the pixels in a region either immediately to the left S_2 or right S_3 , or both.

Accordingly, for regions S_1 and S_4 we set the minimum difference threshold to just 0.05 in order to count virtually all pixels that are darker than p , whereas for regions S_2 and S_3 we set this value to 0.25 so as to only count pixels that are significantly darker than p . When constructing the enhanced image from the results of the operators $\mathbf{L}_{1\dots 4}$ we combined them to construct an image with a light background and dark features. We wish our result to indicate pixels that are darker than the majority of pixels in the regions S_1 and S_4 above and below, and those to the sides in *either* S_2 , S_3 , or both. Firstly we generate an image \mathbf{L}_{sides} indicating pixels that are darker than those in one or both of the side regions

$$\mathbf{L}_{sides}(p) = (1 - 0.5\mathbf{L}_2(p))(1 - 0.5\mathbf{L}_3(p)).$$

We then combine this with the results of the local comparison operators for the regions above and below to determine the enhanced image

$$\mathbf{I}_{enhanced}(p) = (1 - 0.9\mathbf{L}_1(p))(1 - 0.9\mathbf{L}_4(p))(1 - 0.9\mathbf{L}_{sides}(p)),$$

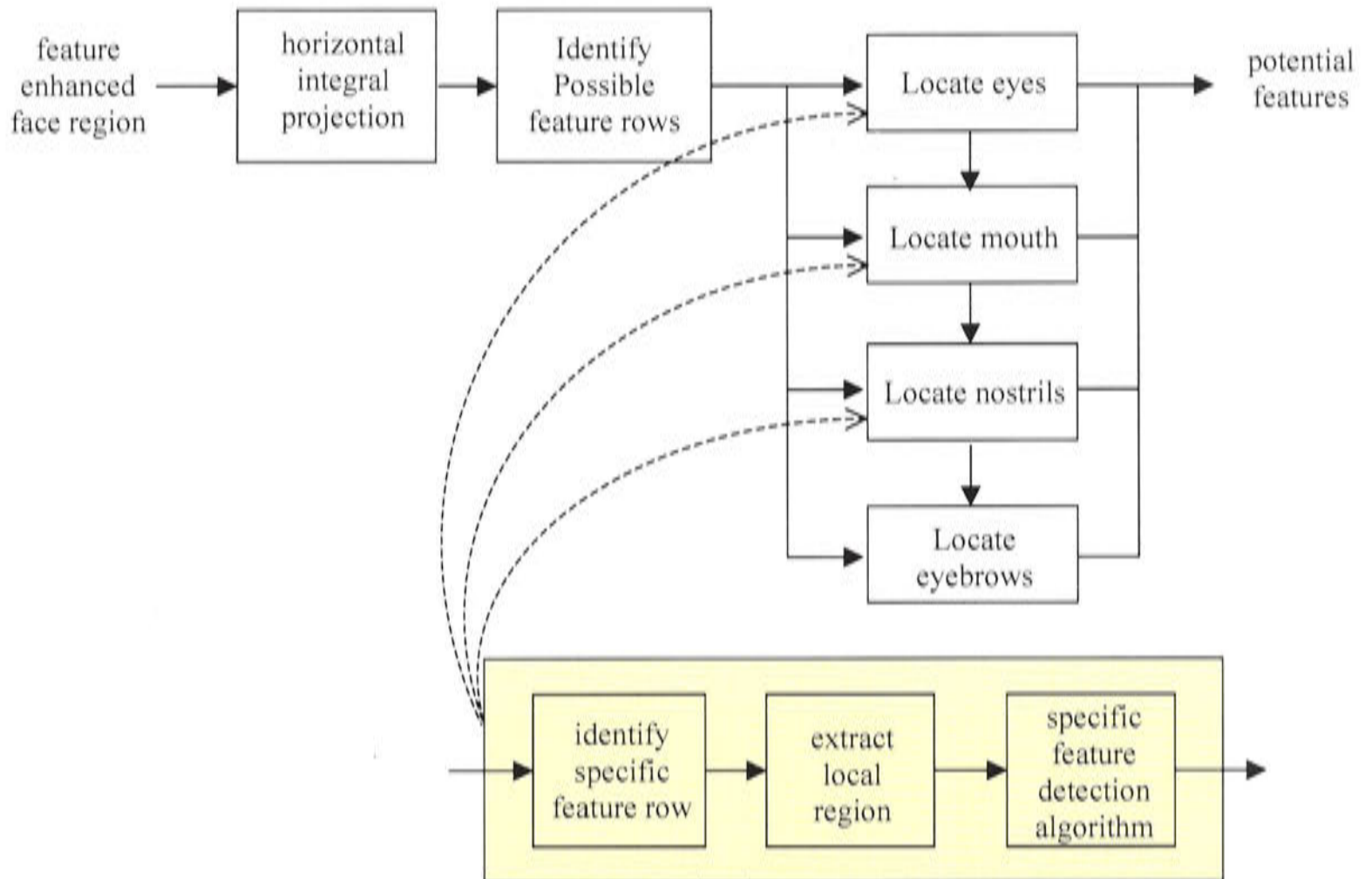


Figure 5.10: Procedure for locating facial features.

An example of the resulting enhanced image is shown in Figure 5.8. Note how the facial features are more clearly defined than in the re-scaled image and are well suited to detection via integral projection.

5.3.4 Classifying Facial Features

Integral projection (Kanade, 1973), described in Chapter 2, forms the basis of the feature detection phase. Integral projection is useful for detecting features whose intensities stand out from the background and have a strong horizontal or vertical aspect. Note that in the procedure described here we take the integral projection of the negative of the enhanced image so that the features stand out as maxima. The raw integral projections are smoothed with a Gaussian to remove high frequency noise. We define a $1 \times (2r + 1)$ Gaussian vector \mathbf{g}_1 with standard deviation $r/3$ which we use for this smoothing.

Figure 5.10 outlines the feature detection process. We take the vertical integral projection of the negative of $\mathbf{I}_{enhanced}$. The integral projection is then smoothed by convolution with the Gaussian vector \mathbf{g}_1 , and the five highest local maxima are identified as potential feature rows. Figure 5.11 illustrates the location of the

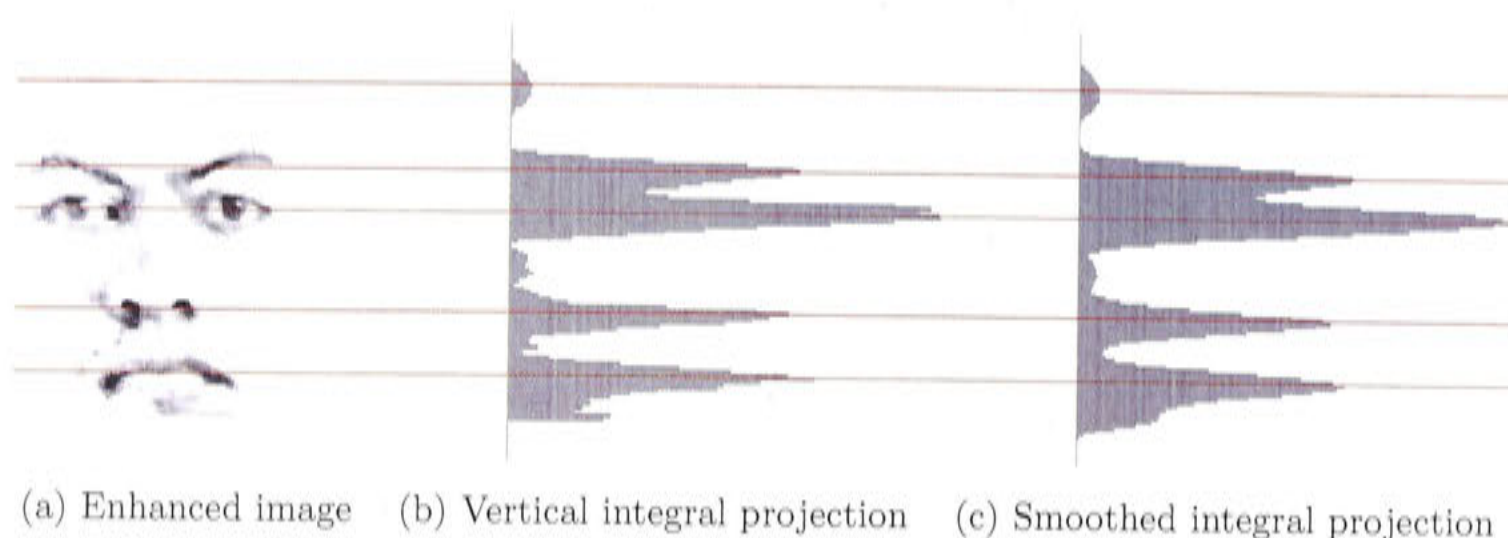


Figure 5.11: Process for locating feature rows using integral projection. The feature rows are indicated by the coloured lines.

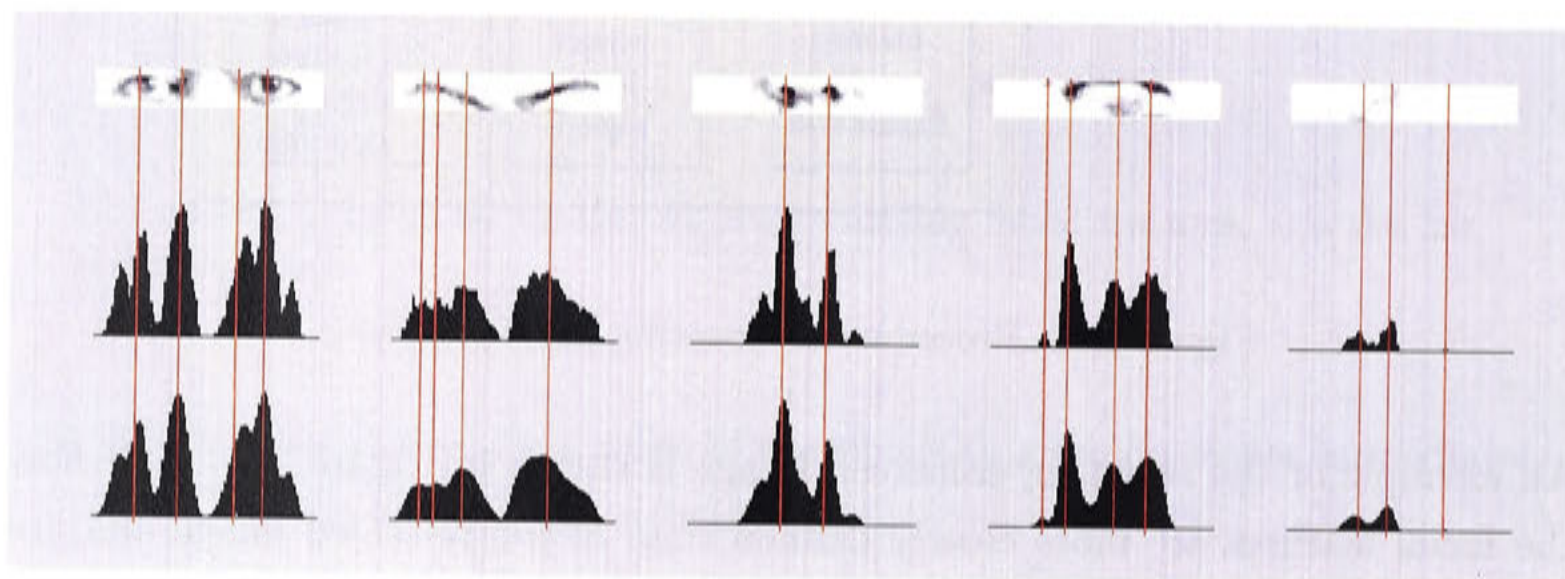


Figure 5.12: Process for locating feature columns within each feature row using integral projection. The feature columns are indicated by the coloured lines. The top row shows sections of the enhanced image on which the integral projection is performed, the middle row shows the result of the integral projection, and the bottom row shows the integral projection after smoothing.

feature rows from an example enhanced image.

Each feature row is then processed as shown in Figure 5.12. Horizontal sections $5r$ high are taken across the face region centred at each potential feature row, and the vertical integral projection of the negative of these horizontal sections is calculated. This is smoothed with g_1 , and the four highest local maxima are taken as potential feature columns within that feature row. Thus we have a set of potential feature rows each with a set of potential feature columns.

Next we attempt to locate the eyes, mouth, nostrils and eyebrows sequentially. If the eyes or mouth are not found, the face region is declared invalid and no face

is detected in this frame.

Each feature location process follows the procedure outlined in the inset in Figure 5.10. Firstly the relevant feature row is determined, local search regions are extracted for the left and right features, these are re-scaled to optimise the intensity distribution, and then a feature detection algorithm specific to the feature type is employed. The only variation on this is the procedure for eyebrow detection where a feature row is not identified prior to locating the eyebrows. Instead the eye locations are used to determine where the eyebrows are expected to be found.

Locating the Eyes

We know the eyes will be located in the vicinity of the blink points, so we insist that the eye feature row not be more than $2r$ above or below the blink points, and that it must have potential feature columns within $2r$ to the left or right of the blink points. If no feature row satisfies the criteria than the algorithm stops examining this frame, not having found a face.

Shadows, creases, makeup or low eyebrows in the region immediately above each eye can cause minima in the integral projections used to place feature rows and columns. However, the cheek area below the eyes is typically devoid of features. Therefore, if there is more than one feature row with appropriate feature columns within $2r$ of the blink points we chose the row closest to the bottom of the image as the eye feature row.

If a satisfactory eye feature row is found then a region about this row is extracted within which to locate potential eye candidates. This search region is defined as the local horizontal section $5r$ high centered about the eye feature row, and the left and right halves of this region are examined to locate the right and left eyes respectively.

Figure 5.13 shows a closeup view of an eye, showing how the pupil and the iris appear as a dark blob with the lighter regions of the sclera to the left and right. Correspondingly our algorithm looks for both a dark iris/pupil blob with light sclera regions on either side.

Figure 5.14 shows the method used to locate potential positions for each eye. Information from the local intensity re-scaled image and the local feature-enhanced

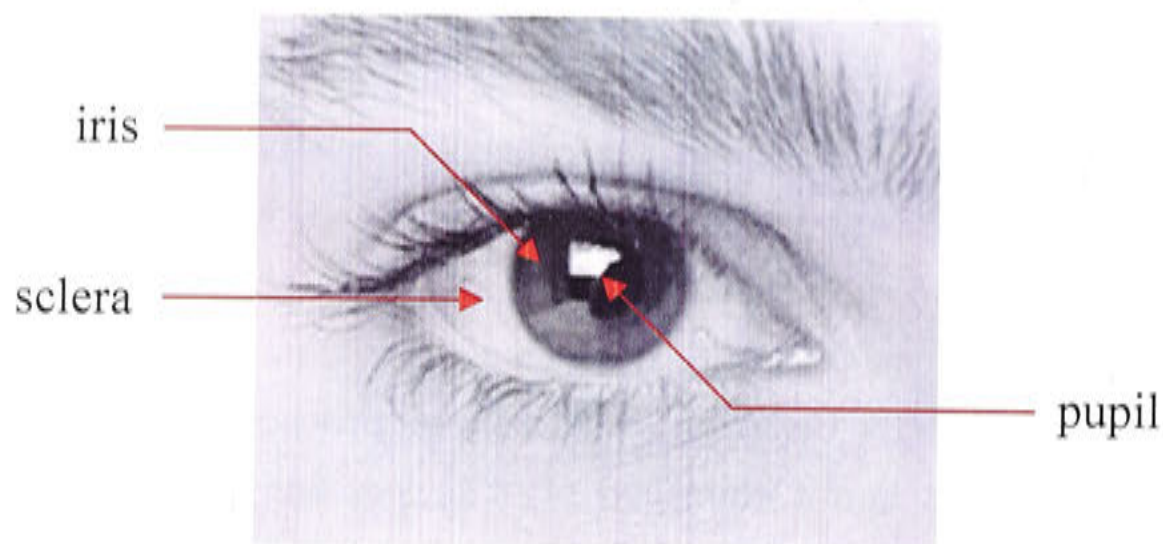


Figure 5.13: Closeup view of a human eye.

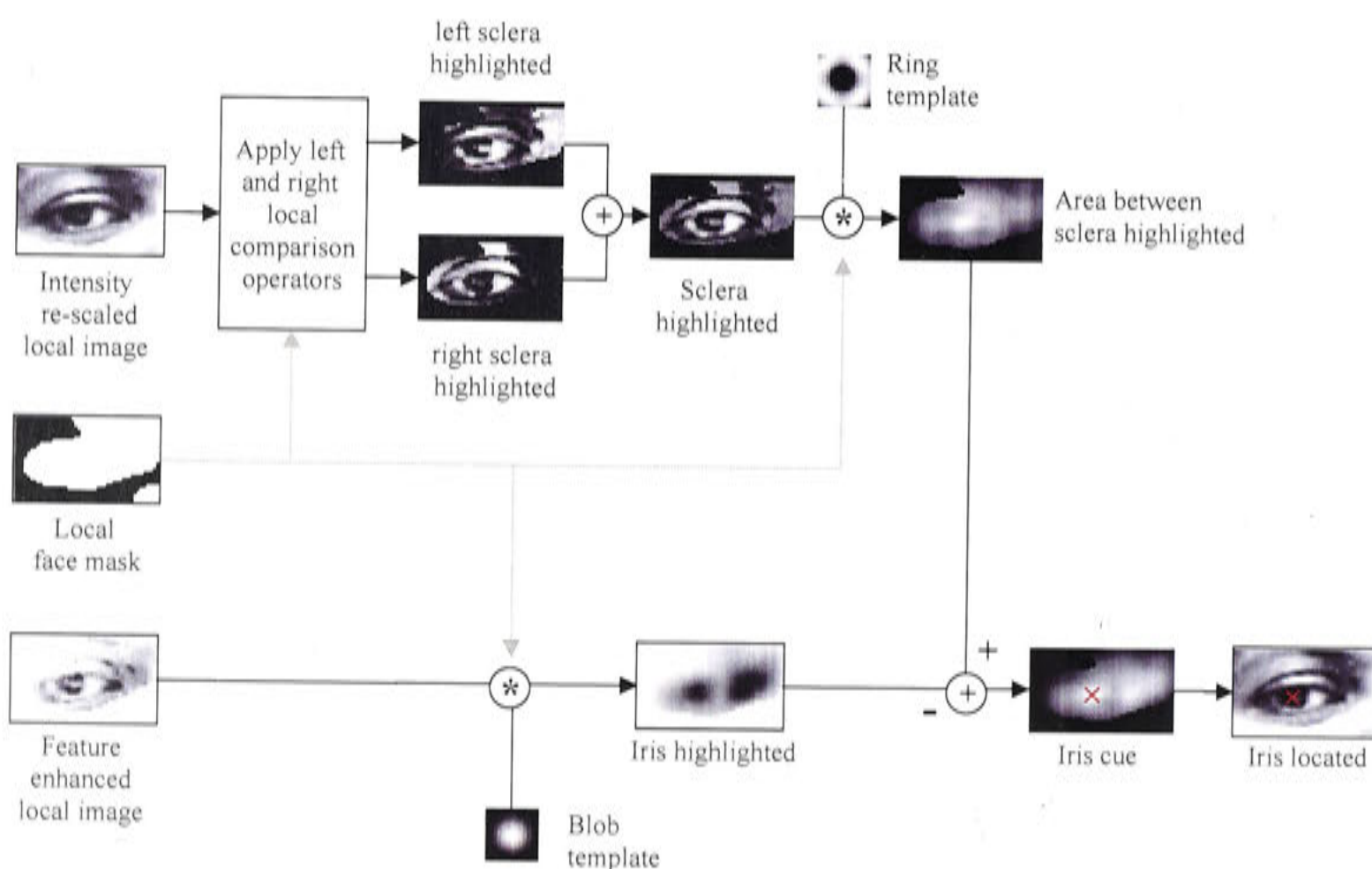


Figure 5.14: Process for locating eyes.

image are used. Only points that pass through the local face mask are considered as possible eye locations, so the local comparison operators and convolutions are only calculated for these points.

The sclera is highlighted in the re-scaled local image by applying a local comparison operator. The operator is applied only at points that pass through the face mask, and uses two $r \times 2r$ regions centered $0.75r$ to either side of the pixel of interest. Figure 5.15 shows one such region used for highlighting the right sclera.

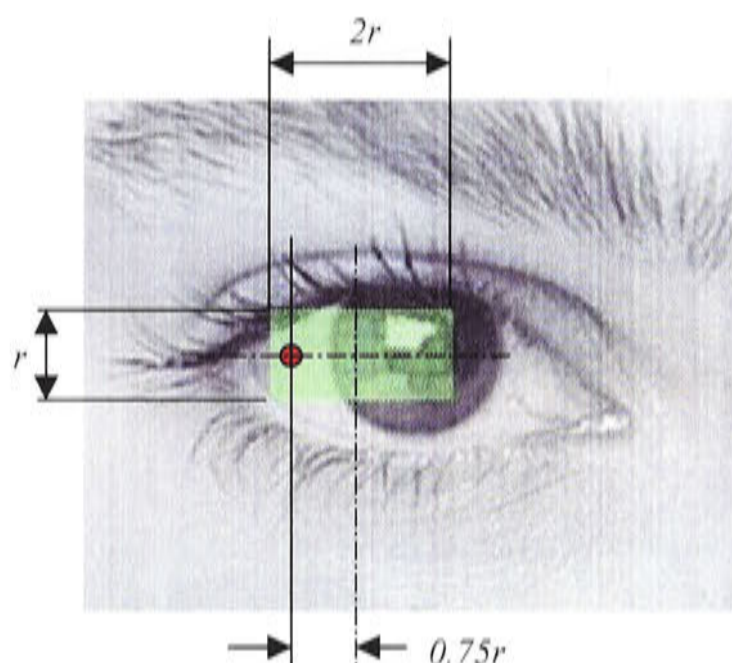


Figure 5.15: Region used by local comparison operator for highlighting right sclera, r is the iris radius.

The resulting image is convolved with a ring template to highlight points that are surrounded by lighter points. This ring template is designed to highlight regions that have sclera-like light regions on the periphery of a iris-size circle, accordingly the ring template chosen is a blurred ring of radius r .

The dark blob of the iris is searched for in the feature enhanced image by convolution with a Gaussian kernel with $s = r/2$.

The results of both the eye-white and iris-blob detectors are normalised so their maximum values are unity. The iris location is then identified as the maximum point in the sum of these two images.

Once both eyes have been found it is verified that the interpupillary distance is realistic, and lies within the range determined in Section 5.2. Also, since the face region was rotated so that the blink points were aligned with the horizontal we expect the eyes to remain closely aligned to the horizontal, furthermore if the line of the prospective eye locations deviates significantly from the horizontal it is indicative that the movements detected were not in fact blinks, therefore it is unlikely that the eyes have been correctly found. To permit some deviation between eye locations and the original blink locations we allow an error of up to 10 degrees between the line joining the eyes and the line joining the blink locations. If these conditions hold the eye locations are accepted and algorithm moves on to locating the mouth.

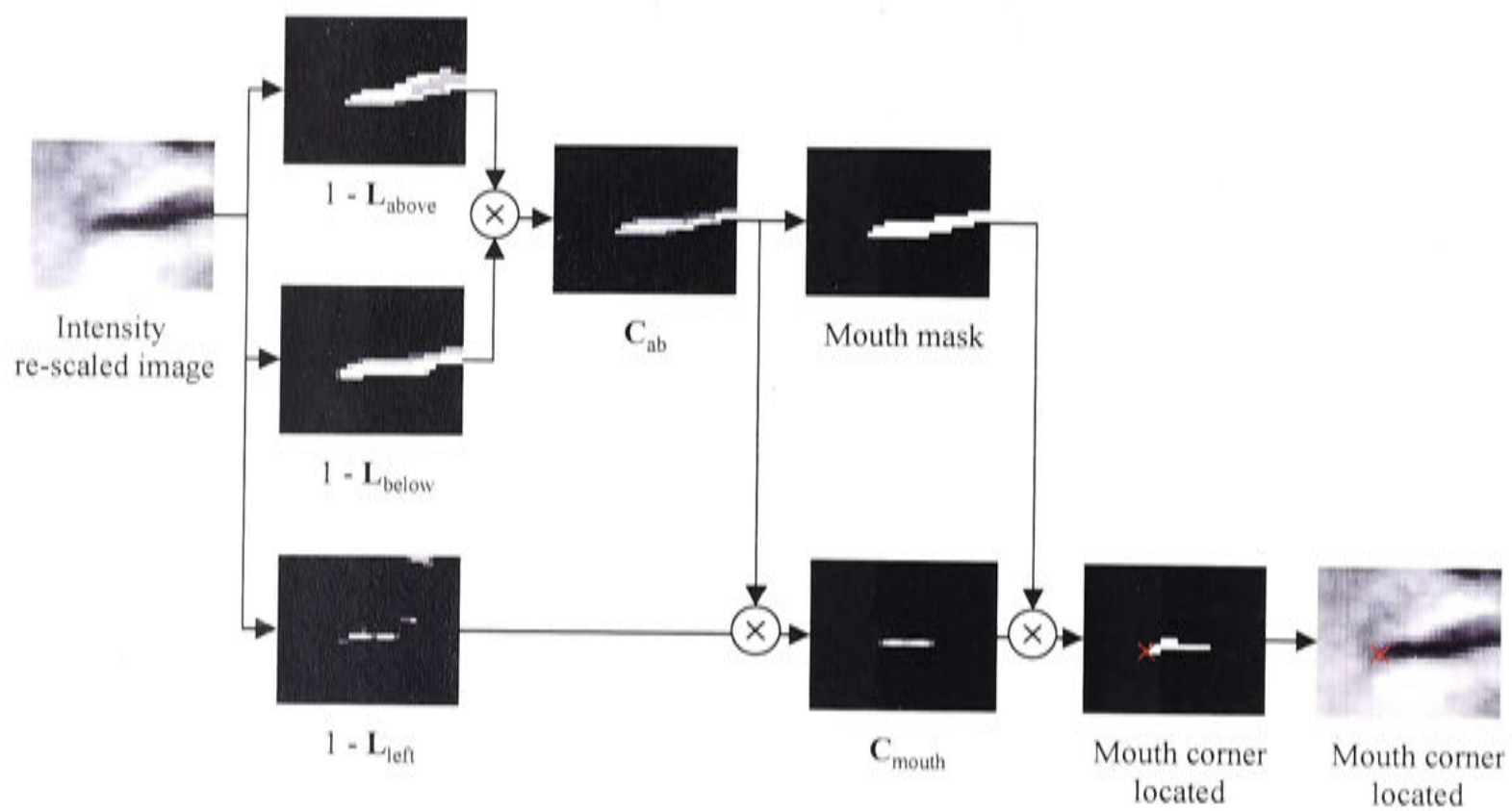


Figure 5.16: Process for locating mouth corner.

Locating the Mouth

The mouth is a dominant feature in the face. It stands out strongly in the enhanced facial image, and with its long, predominately horizontal, shape we expect its location to correspond to a high value in the vertical integral projection of the enhanced image. Subsequently, the mouth row is identified as the feature row located beneath the eyes with the highest integral projection.

From the facial model developed in Section 5.2 we know that the mouth row must be at least $0.67d$ below the eye row for the face to be considered valid, where d is the interpupillary distance. If the mouth row does not meet this criteria then no face is detected in the current frame.

If a valid mouth row is found we then proceed to look for the mouth corners. From the model in Section 5.2 we know the mean mouth corner locations are centred about the vertical axis of the face and slightly closer together than the eyes ($0.1d$ closer towards the centre of the face), furthermore the lateral position of the mouth corners is within $0.3d$ of the mean. Thus we search for mouth corners in $5r \times 0.6d$ regions centered on the mouth row, at the mean lateral mouth corner location (the height of this region is not critical so long as it is large enough to accommodate some variation of the mouth corners from the mouth row).

The process for detecting a mouth corner is shown in Figure 5.16. A mouth corner

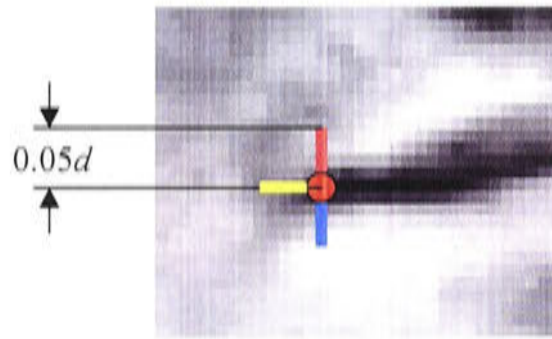


Figure 5.17: Regions for the three local comparison operators used to enhance the appearance of the mouth.

will appear darker than the skin above, below and to the side of the mouth, and this can be utilised by applying a local comparison operator to highlight potential mouth edges. First each region is re-scaled so its intensity is spread over $[0, 1]$. Local comparison operators are used to highlight regions that have lighter regions above and below them, and towards the outside of the face, giving \mathbf{L}_{above} , \mathbf{L}_{below} and \mathbf{L}_{side} . The sets used to calculate the local comparisons are one-dimensional lines of points extending $0.05d$ in the direction of interest away from the point under consideration as illustrated in Figure 5.17.

The output of the local comparison operators \mathbf{L}_{above} and \mathbf{L}_{below} are combined

$$\mathbf{C}_{ab}(p) = (1 - \mathbf{L}_{above}(p))(1 - \mathbf{L}_{below}(p))$$

The result is binarized into positive and zero elements, and connected-component analysis used to identify the largest non-zero region. This then forms a mask \mathbf{M}_{mouth} within which the mouth is assumed to lie.

A cue for the mouth location \mathbf{C}_{mouth} is now determined by combining the results of the local comparison operators, and masking. First the results from all three local comparison operators are combined to form \mathbf{C}_{mouth} ,

$$\mathbf{C}_{mouth}(p) = (1 - \mathbf{L}_{above}(p))(1 - \mathbf{L}_{below}(p))(1 - \mathbf{L}_{side}(p))$$

This is then masked with \mathbf{M}_{mouth} and binarized by converting all positive elements to 1. The edge of the mouth is identified as the non-zero column furthest from the centre of the face. The height of the mouth corners is determined as the height at which \mathbf{C}_{mouth} takes its maximum value in this column.

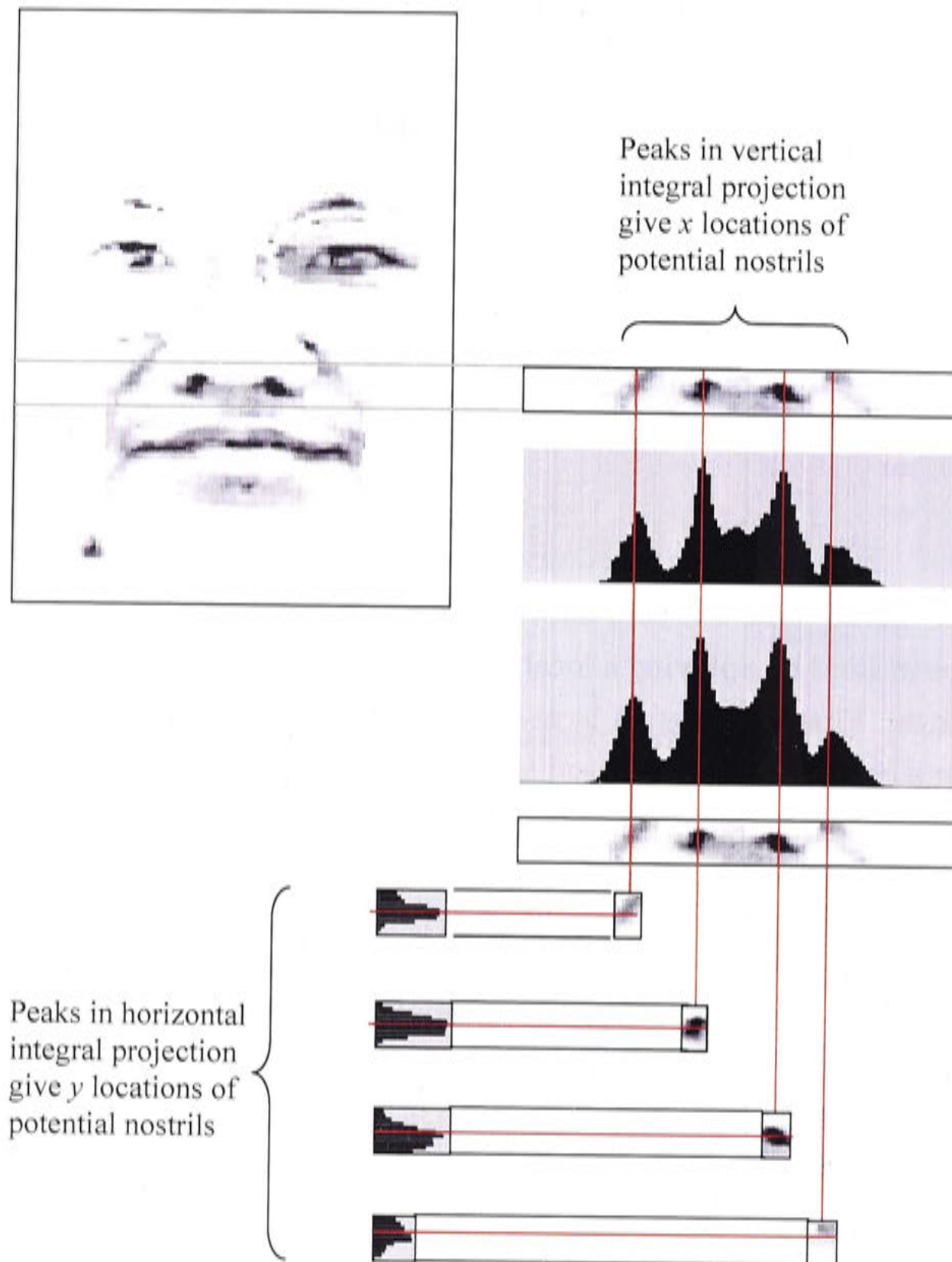


Figure 5.18: Sequence of integral projections used to locate potential nostril locations.

Locating the Nostrils

The nose row is chosen as the most prominent feature row that lies between the eyes and the mouth. In accordance with the face model presented in Section 5.2, if the nose row is higher than two thirds of the way between the mouth and eyes then the face is considered invalid, and no further effort is spent attempting to locate features.

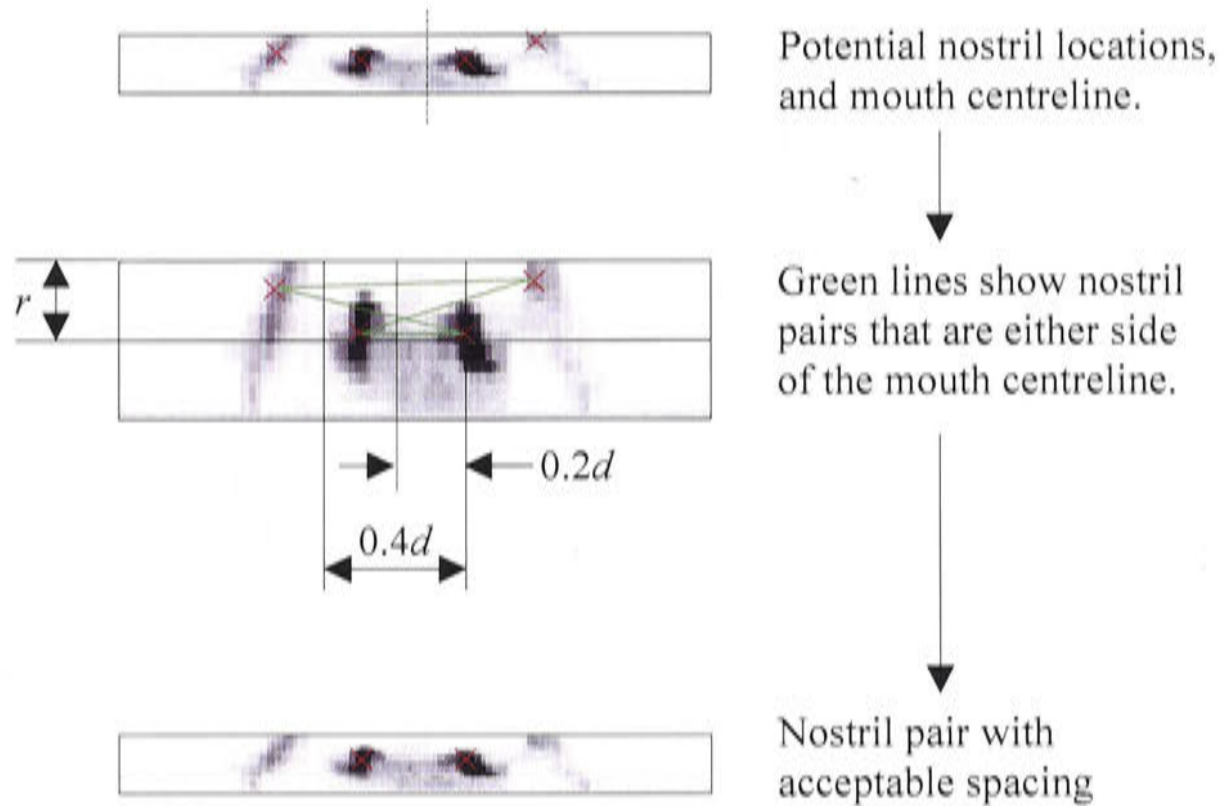


Figure 5.19: Elimination of non-plausible nostril pairs. (Note the centre image has been stretched vertically for clarity.)

If the row height is appropriate we proceed and locate a number of possible nostril locations. The process is illustrated in Figure 5.18. A narrow region $2r$ high and centred about the nose row is extracted from the feature enhanced image. Maxima in the vertical integral projection (of the negative of this narrow region) are used to locate the x coordinates of potential nostril locations. Then small $2r \times r$ regions are extracted around each potential nostril location and maxima in the horizontal integral projection (of the negative of these regions) is used to determine the y coordinate of that potential nostril location.

This process gives a number of possible nostril locations. As illustrated in Figure 5.19 consider each possible pair of locations to find plausible pairs that are:

- on both sides of the mouth centre,
- approximately the same height (within r)
- close together (not more than $0.4d$ apart), and
- not too close together (at least $0.2d$ apart).

If there is more than one plausible pair we select the pair whose horizontal locations are most symmetrically spaced about the mouth centre line.

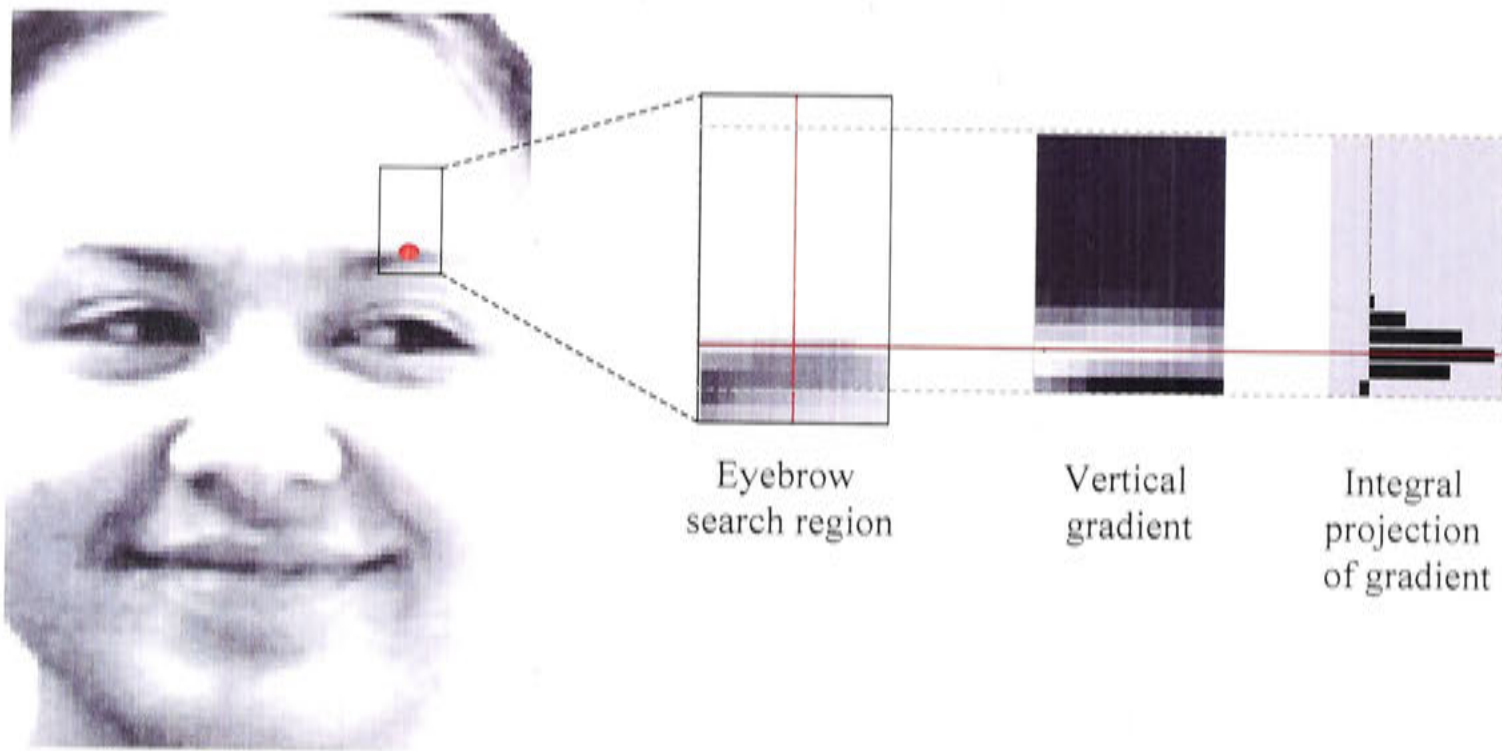


Figure 5.20: Eyebrow detection.

Locating the Eyebrows

The eyebrows are not considered essential features for face detection — indeed some people have very faint eyebrows or none at all — so regardless of whether or not eyebrows are found the face is still considered valid, and the verification process proceeds. However, for the majority of subjects it is a simple matter to locate the eyebrows by looking for a transition from light to dark above the eyes at the base of the forehead.

We already know the approximate location of the eyebrows since the eye locations have been determined. So we proceed to search for the eyebrows in small regions immediately above each eye. The process for locating the eyebrows is shown in Figure 5.20. The eyebrows are searched for in the re-scaled face image in $4r \times 2r$ regions centred $4r$ above each eye. The intensity of these regions are re-scaled so they range within $[0,1]$, and then the vertical gradient is determined via convolution with

$$\mathbf{k} = [-1, -1, 0, 1, 1]^T$$

Vertical integral projection is used to identify the peak in the gradient which signifies a transition from light to dark moving downwards from the forehead towards the eye. This peak is assumed to correspond to the eyebrow, thus the y -coordinate of the eyebrow is found, the x -coordinate is taken to be the same as for the eye.

The procedure is performed for both eyebrows and the resulting eyebrow locations are checked to ensure that their heights do not differ by more than r (the radius of the iris). If heights differ by more than this amount the eyebrows are considered too crooked, and they are declared invalid.

5.3.5 Verify Face Topology

The human face has a specific arrangement of features that is universal across all people. We can verify that the features found by our algorithm are arranged in a face-like configuration by checking that they satisfy the following rules:

- Relative horizontal positioning:
 - Nostrils must not be centred more than $0.5d$ from the eye centre line.
 - The mouth must not be centred more than $0.15d$ from the eye centre line.
- Relative vertical positioning:
 - Nose must be located below one third and above five sixths of the way from the eyes to the mouth.
- Orientation of pairs:
 - The lines joining left and right pairs of features (eg. left and right eyes) must be within 15 degrees of the horizontal.

If the features fail to satisfy these criteria then it is highly unlikely that the features represent a face, so the face is declared invalid and no face is found in the current frame.

5.3.6 Checking Similarity of Feature Pairs

The natural bilateral symmetry of the human face means that all of the feature pairs located by this system should be approximately mirror images of each other. Figure 5.21 shows a face region with the detected features marked with crosses. Small $2r \times 2r$ regions are marked around the features on the left hand side of the figure, for each feature this regions is flipped about the vertical axis and

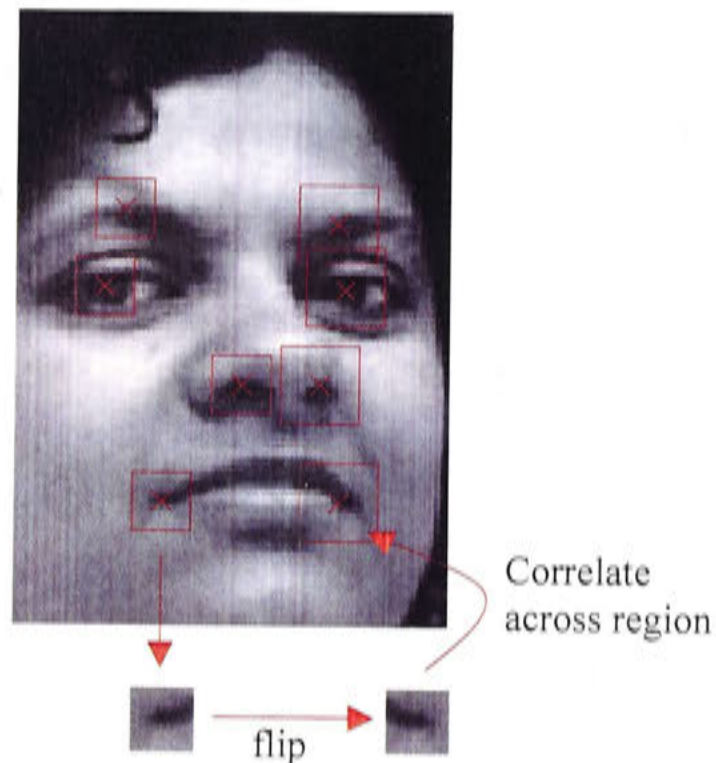


Figure 5.21: The similarity of symmetrically opposite features is verified.

correlated across the $2.5r \times 2.5r$ region surrounding the other half of the feature pair (using normalised cross correlation) to measure the symmetric similarity of the feature pair. The figure illustrates this procedure for the mouth feature pair. We require the mean similarity of all feature pairs to be greater than 0.5 for a set of features to register as a valid face.

5.4 Performance of System

5.4.1 Implementation

The system was implemented and tested in Matlab 5.3 on a standard 600MHz Pentium III. In order to verify the suitability of the algorithm for realtime applications the Matlab flop counter was used to estimate the average number of floating point operations required per frame. The tests were run on sequences of 240×360 grey scale images. The average computational requirement per frame varies depending on how far the algorithm progresses towards verifying the presence of a face. Table 5.1 shows the average number of megaflops required by different stages of the algorithm.

Table 5.1: Estimated Computations per Frame

Stage	Mean computation (megaflops)
Blink detection	6.0
Extract and enhance face candidate	6.0
Classify features	1.6



Figure 5.22: Snapshots of a sequence. Regions of motion are indicated in yellow and blink-like motion is indicated in orange.

5.4.2 Detection Performance

Figure 5.22 shows a number of snapshots of the system in operation. The Figure shows spurious blink-like motion (a) where movement of the eyebrows is incorrectly detected as a blink, however, a face is not detected for these false eye locations since the facial topology is incorrect. In this same frame movement of the eyes is correctly classified as non-blink-like motion since the motion region over the subject's left eye is too small. In frame 32 (b) the subject turns her head and the system does not mistake this gross head motion as blink-like. Likewise the background motion in frame 89 (c) does not distract the system. A blink is detected in frame 102 (e) and so a face is searched for in the frame 0.1 seconds prior to the blink (d). Features are detected and the facial topology is verified



Figure 5.23: Results of the system on a range of subjects. The first column shows the raw image where eye motion is detected, the second column show the motion regions, the third column shows the enhanced facial region extracted, and the fourth column shows the features detected.

giving the final feature locations (f).

The system was tested on a 16 people with a wide range of facial appearances and skin tones, the image sequences were captured both indoors and outdoors and included several subjects wearing eye glasses. Correct facial feature locations were determined in all but one case where the system failed to find the face.

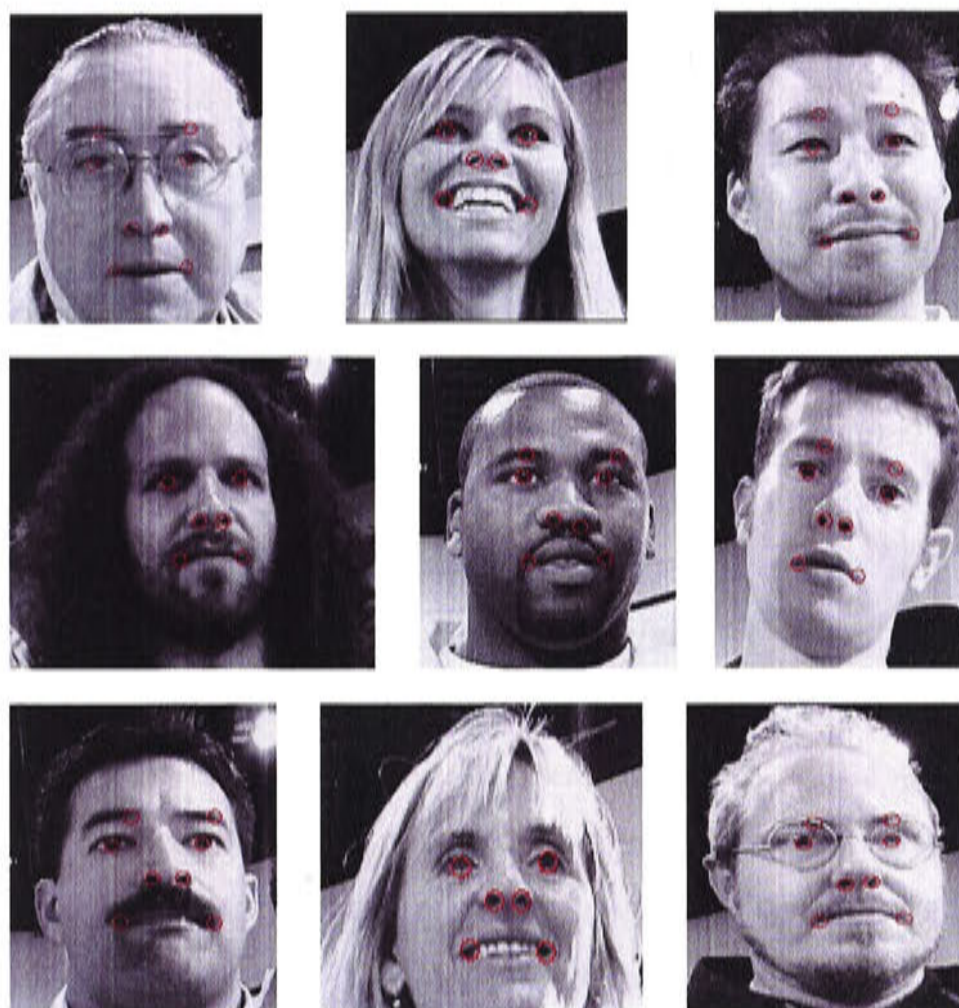


Figure 5.24: Examples of the Seeing Machines implementation of the system in operation, courtesy of Seeing Machines.

There were no false detects. Figure 5.23 shows several examples of the system successfully detecting faces in video sequences. The complete sequences, together with several additional examples, are contained on the CD-ROM enclosed in this thesis.

The robustness of the system comes from its strong ability to reject false matches. Whilst this is achieved at the cost of rejecting some true matches, since the system is operating on a continuous and on-going image sequence, the occasional false rejections can be tolerated. False rejections do nothing to disturb the operation of the system, and searching will continue as the system attempts to locate the target in subsequent frames.

5.4.3 Seeing Machines System

The system has subsequently been implemented in C++ by Seeing Machines to run in realtime. The performance of this realtime system clearly demonstrates the robustness of the method, it has been tested on over 250 subjects and has

demonstrated a 97% success rate for detecting faces and facial features. Figure 5.24 shows several snapshots of the system in operation.

5.5 Summary

This chapter has described a face registration system capable of verifying the presence of a face and performing automatic detection of facial features. The system uses motion information to detect blinks, indicating possible eye locations and an associated face candidate. Filtering methods are used to enhance the appearance of potential facial features in the face candidate region and integral projection is used to estimate the individual feature locations. The system is able to locate the eyes, mouth corners, nostrils and eyebrows of subjects in a monocular, monochrome image sequence. The feature locating and face verification process is governed by the anatomical constraints of a human face. The purpose of this face registration procedure is to both automatically verify the presence of a face and detect facial feature points for face tracking.

The performance of the system has been demonstrated on numerous image sequences, and the algorithm has been adopted into a commercial face detection system, which has seen substantial testing on hundreds of subjects.

Chapter 6

Face Tracking

IN the previous two chapters we considered the problems of face localisation and face registration. The final step towards enabling a computer to see the face is face tracking. This involves tracking both rigid and deformable facial features in order to fully characterise both the pose (3D position and orientation) of the head, and describe the locations of facial features relative to the head. By *rigid* facial features we mean features that are rigidly attached to the head — such as the eye sockets, nose and ears — as distinct from *deformable* features such as the mouth and eyebrows that change shape and move relative to the head. To achieve this we track the 3D pose of the head using predominantly rigid facial features, and then consider tracking the locations of deformable features relative to the head.

The mouth is the most important deformable facial feature for Human Computer Interaction, and it is a challenging feature to track. We restrict our consideration of deformable feature tracking to tracking the mouth. It is, however, feasible to adapt the approaches described here to tracking other deformable features such as the eyebrows. Two case studies in lip tracking are presented. The first is a monocular lip tracker that tracks the height and width of the mouth, and the second is a stereo lip tracking system. Our stereo system is the first to use stereo to directly recover the full 3D shape of the mouth. Both systems track unadorned lips, and do not require subjects to wear lipstick or other cosmetic aids to enhance the appearance of the mouth. The systems operate on grey-scale images and run in conjunction with a head tracker to enable robust performance through a range of head poses.

Section 6.1 addresses the problem of tracking deformable features whose appear-

ances change significantly as they elastically deform during a tracking sequence. Section 6.2 presents the monocular lip and head tracking system, and Section 6.3 presents the stereo lip tracker capable of reconstructing the 3D shape of the outer lip contour. Finally, Section 6.4 concludes the chapter with a summary of our findings.

6.1 Adaptable Templates

Template tracking is a well established method for tracking features in an image sequence. A template containing a sample of the feature to be tracked is correlated across a search region, quantifying how similar the template is to different parts of the search region, and the target is located at the point with the highest correlation. Numerous different correlation measures can be used (refer to Chapter 2) but the most common method is Normalised Cross Correlation (NCC). Denoting the template as \mathbf{I}_1 , the search window as \mathbf{I}_2 , and summation over the window as $\sum_{(u,v) \in W}$, the NCC of \mathbf{I}_1 and \mathbf{I}_2 is given by

$$\text{NCC}(\mathbf{I}_1, \mathbf{I}_2) = \frac{\sum_{(u,v) \in W} \mathbf{I}_1(u, v) \cdot \mathbf{I}_2(x + u, y + v)}{\sqrt{\sum_{(u,v) \in W} \mathbf{I}_1(u, v)^2 \cdot \sum_{(u,v) \in W} \mathbf{I}_2(x + u, y + v)^2}}$$

NCC has the advantage that it is invariant to linear changes in intensity, that is

$$\text{NCC}(\mathbf{I}_1, \mathbf{I}_2) = \text{NCC}(\mathbf{I}_1, k\mathbf{I}_2 + l)$$

where k and l are scalar constants, which gives it some robustness to changes in lighting. Figure 6.1 shows an example of template matching with NCC. Here a template is chosen at the mouth corner in (a). When correlated across the original image the NCC peaks at the mouth corner correctly locating the feature as shown in (b). Applying a linear change in intensity to the search image in (c) has no effect on the resulting NCC with the unadjusted template from (a).

However, whilst NCC can provide robustness to variations in lighting it cannot accommodate features whose shapes deform and change appearance through an image sequence. The changing appearance of deformable features, such as the mouth corners, make standard fixed templates ineffective. For instance, Figure 6.2(b) shows the result of using a mouth corner template defined in frame 1 (Figure 6.2(a)) to try and locate the same mouth corner 222 frames later in the

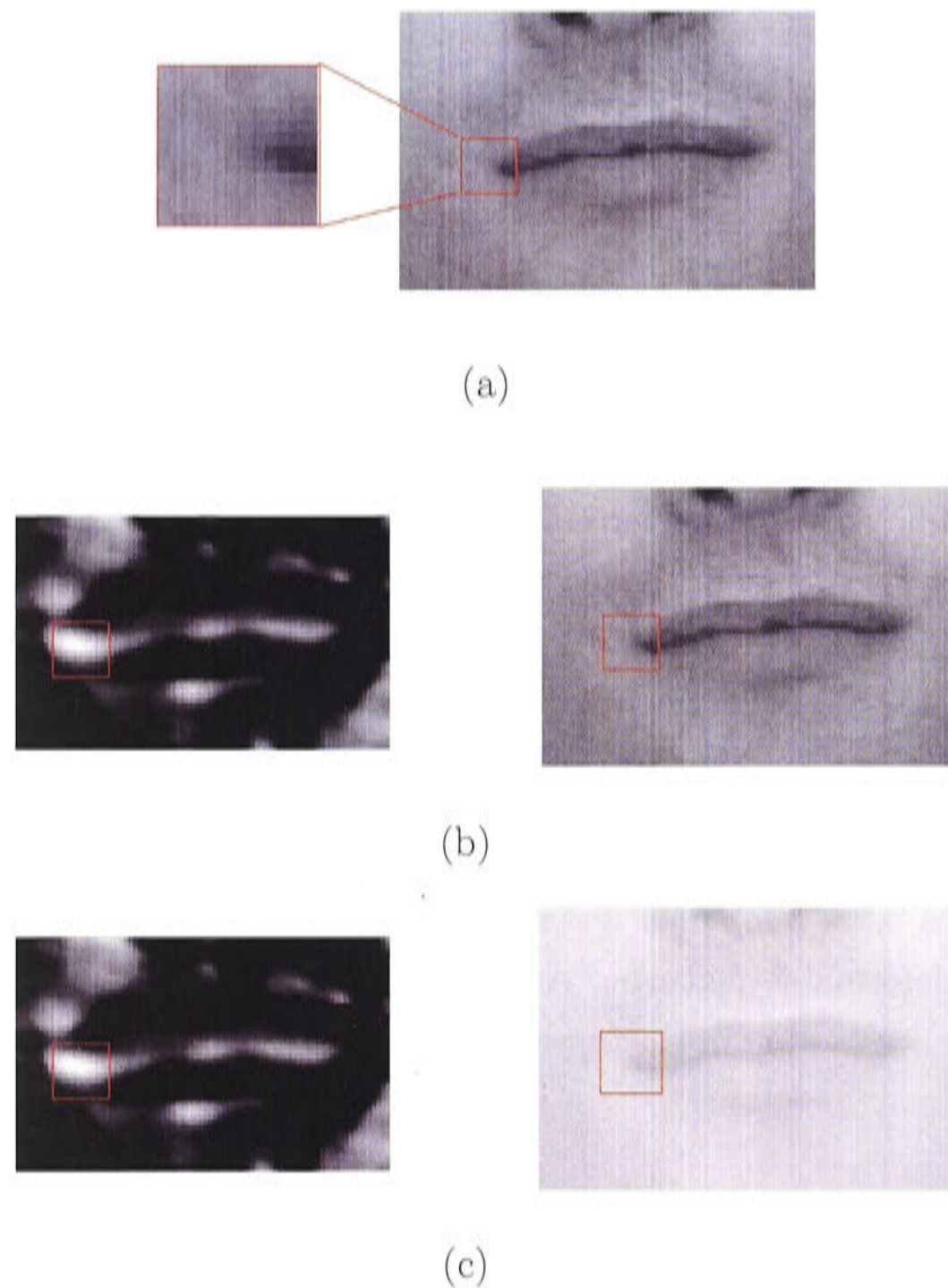


Figure 6.1: Template matching. (a) A feature template is defined. (b) NCC is used to match the template across the image, the maximum of the NCC indicates the location of the best match this is shown by the orange square. (c) NCC with the same template after a linear change in intensity to the image, the resulting NCC is identical to that for the unaltered image.

same image sequence. The feature's appearance no longer closely resembles the template and the NCC peaks at the edge of the mouth cavity rather than the mouth corner, incorrectly locating the feature.

In order for templates to maintain adequate tracking performance whilst tracking a deformable feature it is necessary to dynamically adapt the template to keep it up-to-date with the current appearance of the target. An obvious approach is to update the template each tracking cycle to equal the new feature appearance. There are, however, several problems with this approach. Firstly, if there is ever

an error in the template matching and the new template is chosen off target there is no way for the system to recover from this error. The chance of this occurring can be minimized by only updating the template when the match is good. However, even if there is never any incorrect matching, each frame the template is only located up to a finite degree of accuracy, typically to the nearest pixel, although greater precision can be obtained using sub-pixel placement. Therefore, there will be an error of up to half the template placement precision every frame, i.e., up to half a pixel if sub-pixel placement is not used. With 30 frames every second, over time this small error is likely to accumulate and cause the updated templates to drift away from the features they are supposed to be tracking. Figure 6.2(c) shows an example of this template drift over 222 frames. Here the template was only updated when a “good” match was found ($\text{NCC} > 0.75$), and the final location of the template is a consequence of drift, and not a result of incorrect matching.

To address this issue *adaptable templates* have been developed for the tracking of elastically deformable features. Adaptable templates make use of the initial object appearance as seen in the original templates that were correctly initialised to the true feature locations.

Adaptable templates work as follows: For the k^{th} frame, once the best match is found (and provided the correlation is above a certain threshold), the template $\mathbf{T}_i[k]$ is updated to become the weighted average of the initial template $\mathbf{T}_i[0]$ and the image region $\mathbf{R}_i[k]$ in the new frame that best matches the current template, that is

$$\mathbf{T}_i[k + 1] = \alpha \mathbf{T}_i[0] + (1 - \alpha) \mathbf{R}_i[k], \quad (6.1)$$

where the constant $\alpha \in (0 \dots 1)$ is the *grounding factor* which determines the contribution of the initial template to the new template. $\alpha = 0$ is the case of fully updated templates and $\alpha = 1$ gives standard templates.

Figure 6.2(d) shows the result of tracking the mouth corner with an adaptable template with $\alpha = \frac{1}{3}$. The initial template is chosen as shown in Figure 6.2(a) and adapted each frame according to Equation 6.1. Even though the initial template was taken with the mouth in a neutral position, the adaptable template is able to correctly locate the mouth corner when the mouth is wide open despite the drastic change in the appearance of the feature. Furthermore, always

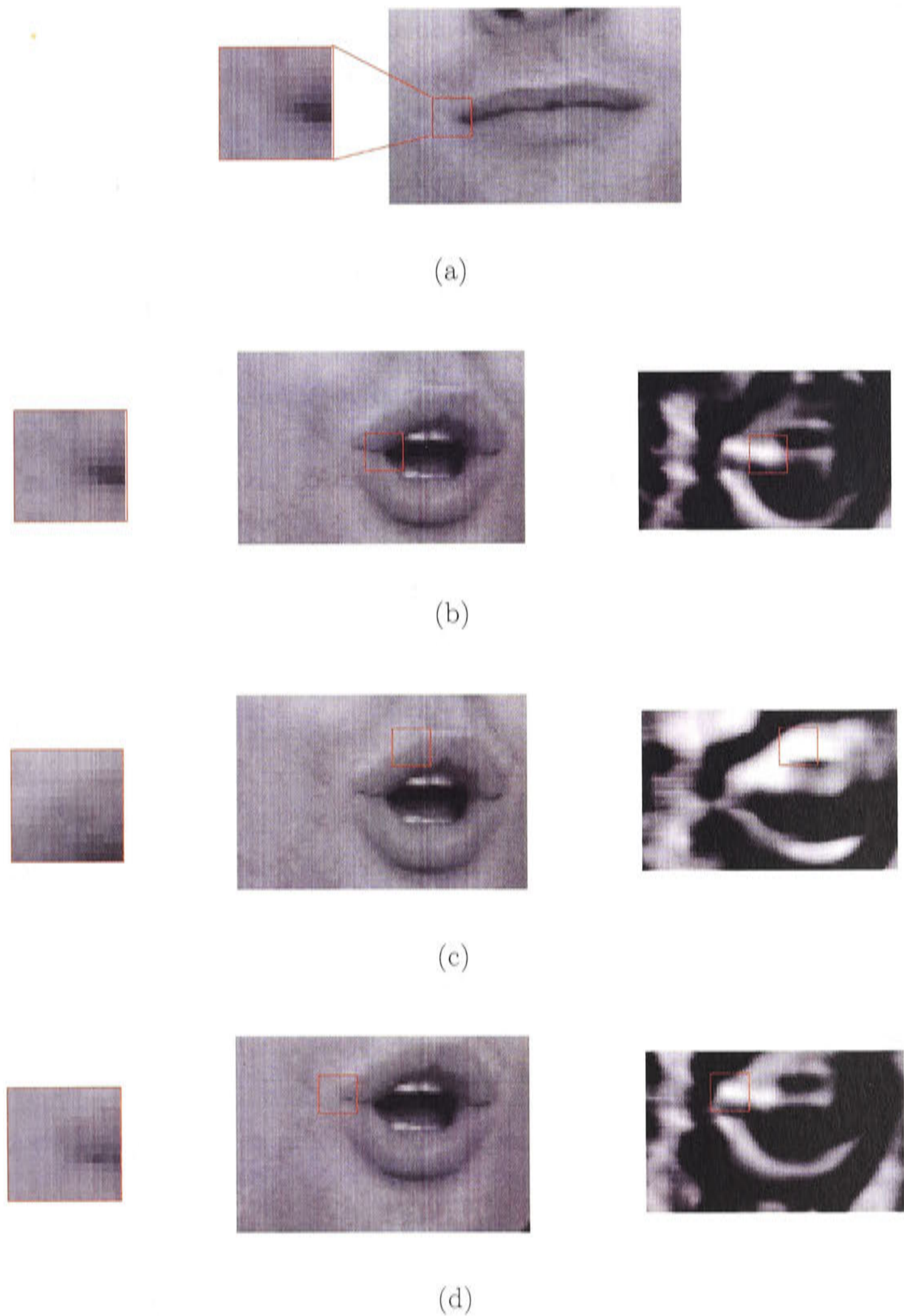


Figure 6.2: Template matching a deformable feature using NCC. (a) Initial template chosen in frame 1. (b) Using initial template in frame 222. (c) Using fully updated template in frame 222. (d) Using an adaptable template in frame 222.

keeping the adaptable template “grounded” with the original feature appearance $\mathbf{T}_i[0]$ prevents the updated adaptable templates from drifting off the features as happened for the fully updated templates.

Adaptable templates are useful for tracking elastically deformable features that change appearance, but regularly return to their original appearance. This makes them especially well suited to tracking deformable facial features such as the mouth corners or eyebrow edges. However, adaptable templates are not limited to tracking deformable features. Consider for instance, that the eye corners are being tracked by a head tracker. Disregarding eye closure these features do not exhibit a significant amount of deformation and are traditionally tracked by fixed templates. However, as the subject moves and rotates his or her head the appearance of these features will change as they are viewed from different angles, and when the subject’s head returns to its initial position the features will appear the same as they did initially. So, although these features are not actually deforming their *appearance* is deforming in an elastic manner, and thus the tracking of such features stands to benefit from adaptable templates.

6.2 Monocular Lip Tracking

This section describes a head and lip tracking system¹ that tracks the head and mouth of a speaker whilst allowing the speaker’s head to move in 3D within a 30 cm³ workspace, and rotate up to 30 degrees away from the camera. During operation the system performs the following tasks:

- Computes the pose information using a 3D face tracking system,
- Tracks the top, bottom and corners of the mouth in the 2D input image,
- Estimates the 3D locations of the top, bottom and corners of the mouth, and
- Determines the mouth dimensions from the 3D mouth information.

In Section 6.2.1 we describe the 3D head tracker used to determine the head pose in each frame. Section 6.2.2 then explains the detection of the mouth features,

¹The system was developed by the author at the University of Western Australia as part of an automatic lip reading project at the Department of Computer Science under the supervision of Dr Eunjung Holden and Professor Robyn Owens.

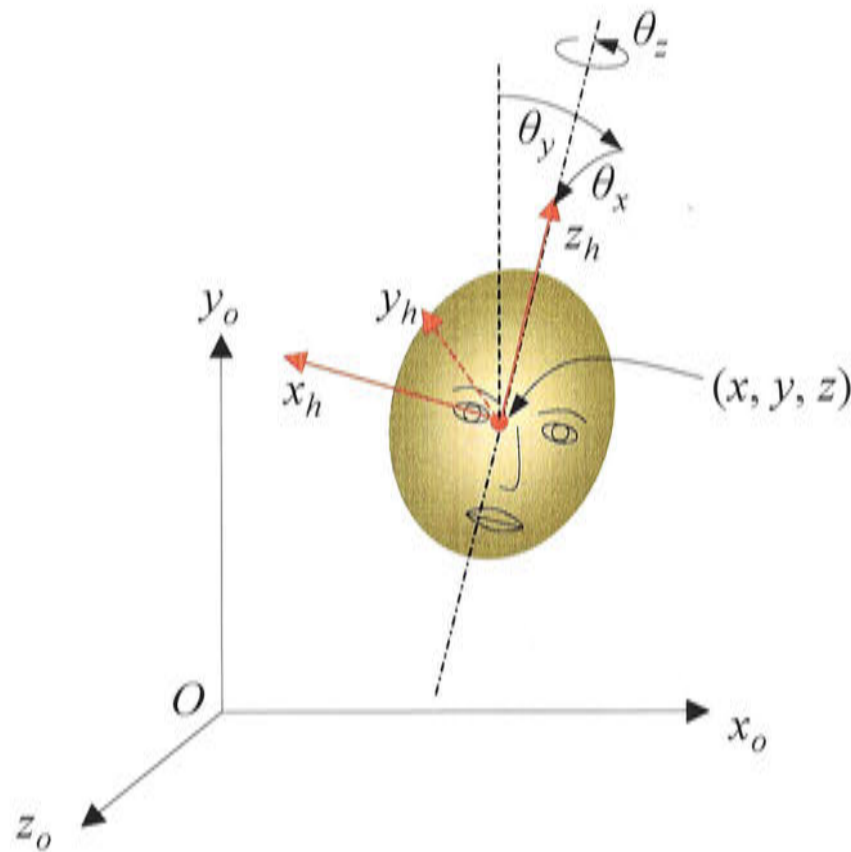


Figure 6.3: 3D pose of a head. Head reference frame shown in orange, and the pose $(x, y, z, \theta_x, \theta_y, \theta_z)$ with respect to the world coordinate frame O indicated.

the estimation of their locations in 3D, and the calculation of the corrected mouth dimensions. Finally, Section 6.2.3 demonstrates the usefulness of this technique with a practical experiment: a subject is tracked whilst he enunciates 5 phonemes “W”, “long-E”, “M”, “short-A”, and “long-O”, displaying the 5 mouth shapes of visual speech while moving and turning his head in 3D.

6.2.1 Monocular 3D Head Tracker

The purpose of our head tracker is to determine the head pose, that is the location and orientation of the head in 3D space. The head is modelled as a rigid body with a reference frame attached, and the pose of the head is defined by a six parameter vector $\mathbf{p} = (x, y, z, \theta_x, \theta_y, \theta_z)$ specifying the Cartesian co-ordinates and rotation of the head reference frame with respect to a predefined world coordinate system. Figure 6.3 shows a schematic of a head with reference frame attached showing the pose of the head reference frame in the world coordinate system.

Our monocular 3D head tracker was based on Lowe’s object tracking algorithm (Lowe, 1991). As discussed in Chapter 2, Lowe’s method provides a model-based approach to determining the pose of a known 3D object. When a 3D object, such as a head, is viewed in an image the locations of its features are a non-

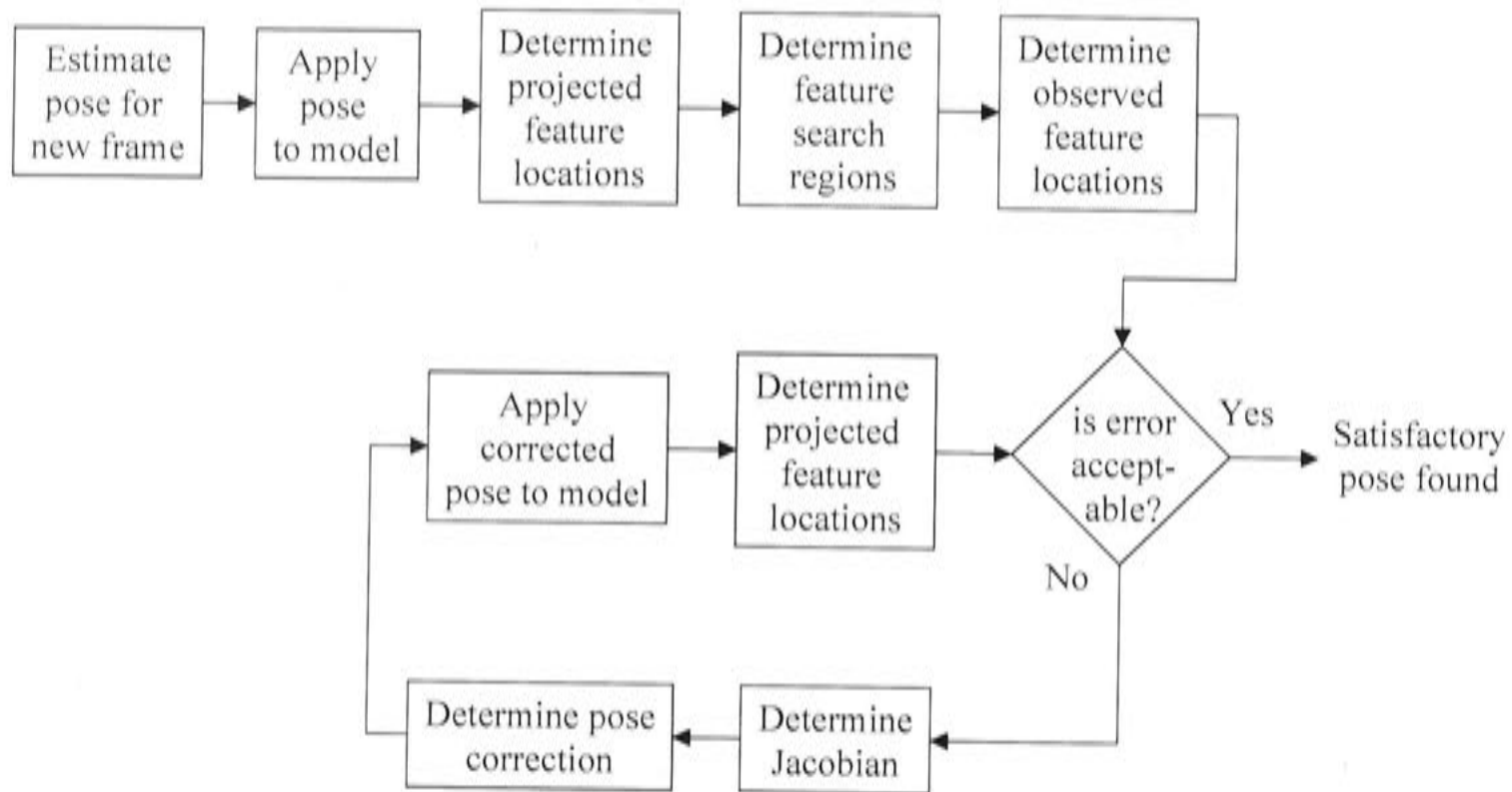


Figure 6.4: The process the head tracker steps through each frame.

linear function of the object's pose. Given an initial guess of the pose, a least squares solution can be achieved iteratively by applying Newton's method to locally linearize the problem. In order to obtain stable approximate solutions in the presence of noise Lowe augmented this minimization by incorporating a model of the range of uncertainty in each pose parameter, together with estimates of the standard deviation of the image measurements, into the procedure. In addition he used the Levenberg-Marquardt method to ensure the solution converges to a local minimum.

Before the algorithm commences it is necessary to initialise the system by identifying feature points to be tracked and provide an accurate 3D model of these points. The 3D model contains the Cartesian coordinates of the features in the head reference frame, and is defined as

$$\mathbf{M} = \left(\mathbf{m}_1 \quad \mathbf{m}_2 \quad \dots \quad \mathbf{m}_n \right),$$

where $\mathbf{m}_i = (x_i, y_i, z_i)^\top$ contain the 3D coordinates of the i^{th} feature point in the head reference frame. A template \mathbf{T}_i is initialised for each feature. Currently features are chosen manually via mouse-clicks, however, the auto-initialisation process described in Chapter 5 is suitable for automatically selecting these features.

For each frame the head tracker goes through the steps illustrated in Figure 6.4. These are discussed in detail in the following section but can be summarised as

follows:

- Estimate the pose of head based on the pose in the previous frame and the average translational and angular velocity over the last 5 frames.
- Apply this pose to the 3D model to determine the 3D location of model points.
- Project these model points onto the image to give projected feature locations.
- Use the projected feature locations to define search regions for the features.
- Determine observed feature locations via template matching across each search window.
- Check if the error between projected model feature locations and observed feature locations is acceptable, if so the current pose estimate is satisfactory, if not proceed to iteratively refine the pose estimate by repeating the following until a satisfactory pose is found:
 - Determine the Jacobian that contains the partial derivatives of the feature locations with respect to the pose.
 - Use the Jacobian to calculate the pose correction factor.
 - Update the pose estimate by adding the pose correction factor and apply the updated pose to the model.
 - Determine the new projected feature locations.
 - Check if pose is satisfactory by checking the error between the new projected feature locations and the observed feature locations.

Pose Estimation

In each new frame the pose is estimated from the pose in the previous frame by adding an offset determined by the estimated translational and angular velocity between the frames. These velocity terms are determined as the average translational and angular velocity over the last five frames. That is, given the poses of the current and previous five frames $\mathbf{p}[t - i]$ for $i = 0 \dots 5$ the pose of the next frame is estimated as

$$\hat{\mathbf{p}}[t + 1] = \mathbf{p}[t] + \sum_{i=1}^5 \frac{\dot{\mathbf{p}}[t - i]}{5}$$

where the pose velocity is determined by the first order approximation

$$\dot{\mathbf{p}}[t] = \mathbf{p}[t] - \mathbf{p}[t - 1]$$

In the first frame an initial estimate of the pose is provided and the pose is assumed not to have change over the previous five frames.

Applying Pose to Model

The pose specifies the six degrees of freedom of the head, namely the translational (x, y, z) location, and orientation $(\theta_x, \theta_y, \theta_z)$. In order to apply these to the 3D model \mathbf{M} we form a translation vector \mathbf{t} and a rotation matrix \mathbf{R} ,

$$\mathbf{t} = (x, y, z)^\top$$

$$\mathbf{R} = \mathbf{R}_z \mathbf{R}_y \mathbf{R}_x$$

where

$$\mathbf{R}_z = \begin{pmatrix} \cos \theta_z & -\sin \theta_z & 0 \\ \sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_y = \begin{pmatrix} \cos \theta_y & 0 & \sin \theta_y \\ 0 & 1 & 0 \\ -\sin \theta_y & 0 & \cos \theta_y \end{pmatrix}$$

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x \\ 0 & \sin \theta_x & \cos \theta_x \end{pmatrix}$$

This pose is applied to the model to determine the new 3D locations of the model points. These are then given by the columns of

$$\mathbf{M}_{new} = \mathbf{R}\mathbf{M} + \mathbf{t}$$

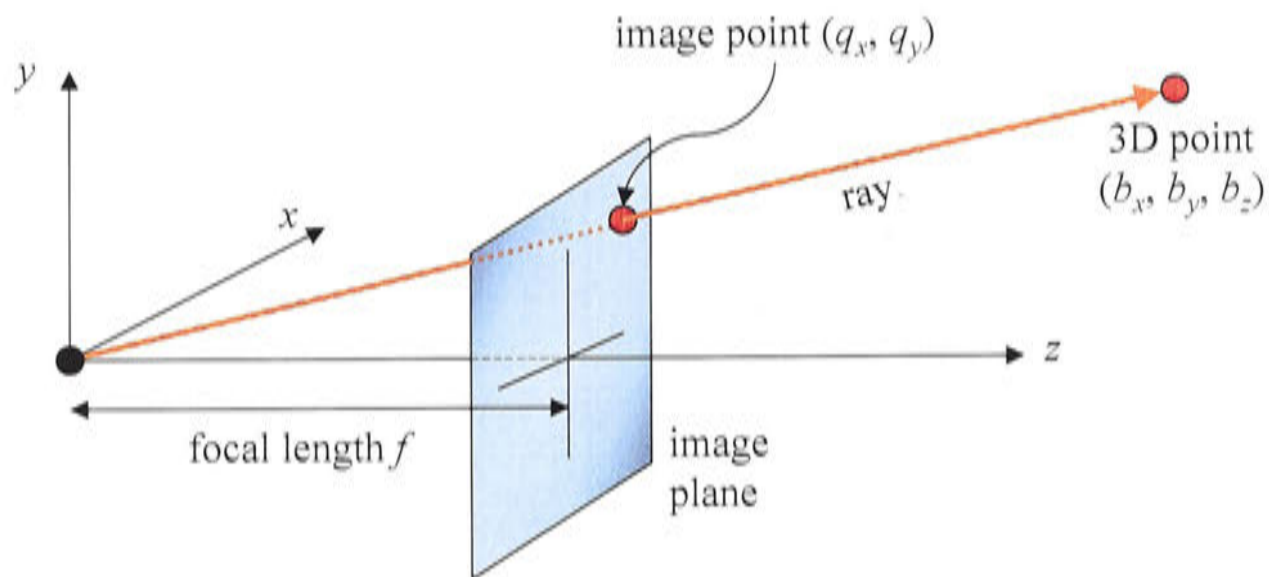


Figure 6.5: Pinhole camera model.

Determining Projected Feature Locations

Once the 3D locations of the model points are determined under the current pose these are projected onto the image plane using the pinhole camera model for projection.

The pin-hole camera model, illustrated in Figure 6.5, performs a true perspective projection of the 3D feature points from the world coordinate frame onto the image plane. A 3D feature point $\mathbf{b} = (b_x, b_y, b_z)^\top$ is projected onto the image plane at

$$\begin{pmatrix} q_x \\ q_y \end{pmatrix} = -\frac{f}{b_z} \begin{pmatrix} b_x \\ b_y \end{pmatrix},$$

where f is the focal length of the camera.

Determine Feature Search Regions

The projected feature locations provide us with an estimate of approximately where the features are likely to appear in the image, based on the estimated pose derived for the current frame. We now define search regions centred at each of these projected feature locations in the belief that the true feature locations will lie within these search regions. The search windows were chosen to be $m \times m$ pixels. Figure 6.6 shows an example face image with the projected feature locations indicated with yellow crosses and the resulting search regions shown as dashed boxes.



Figure 6.6: Face image showing projected feature locations and search regions for $m = 36$.

Determine Observed Feature Locations

The observed feature locations are determined by template matching within the defined search regions. Normalised Cross Correlation is used and the feature templates are updated using the *adaptable template* technique detailed in Section 6.1. This technique is designed to accommodate for the changing appearance of features. Whilst the head tracker uses non-deformable features for tracking, the appearance of the features still change as they are viewed from different angles, so the tracking benefits from using adaptable templates.

The computational load of the template matching is significantly reduced by performing the search in two stages: an initial sparse search to localise the feature, followed by a localised detailed search. Figure 6.7 illustrates this procedure. The sparse search consists of correlating the image with the template centred at every second pixel in every second row and column within a large $m \times m$ search window. The point that returns the highest correlation in the sparse search becomes the centre of the detailed search. The detailed search has a radius of only 2 pixels and the template is correlated at every location in this small local region, the point of highest correlation giving the new feature location.

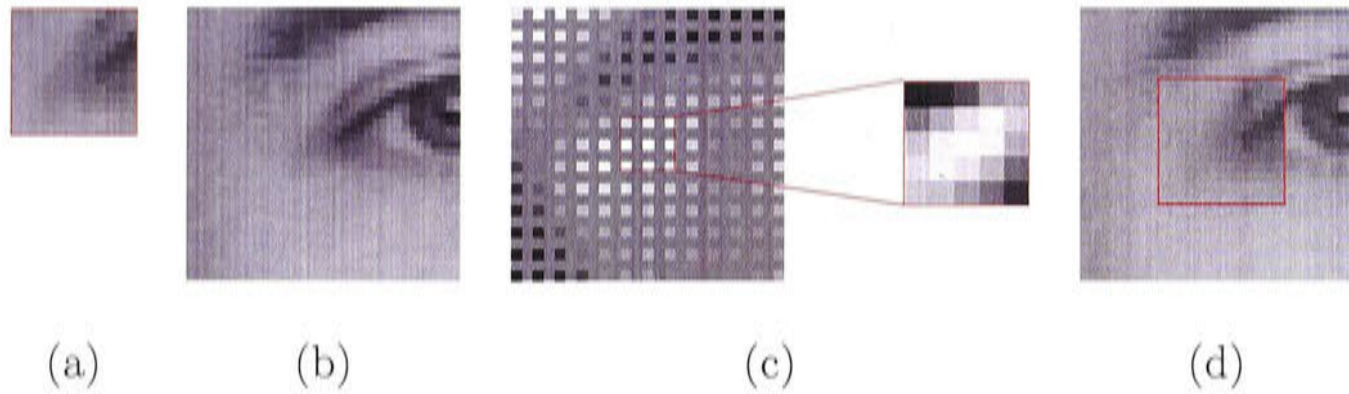


Figure 6.7: Searching procedure using initial sparse search. (a) Feature template. (b) Full search region. (c) Initial sparse search only correlates at every second row and column, a region around the maximum is then identified and searched at full resolution. (d) The template is correctly located to the nearest pixel.

Check if error is acceptable

The observed feature locations provide the target state for our system. The goal of the tracker is to match the projected feature locations with the observed feature locations. We defined the error between these two sets of feature locations as the vector

$$\mathbf{e} = \begin{pmatrix} |q_{x1} - o_{x1}| \\ |q_{y1} - o_{y1}| \\ |q_{x2} - o_{x2}| \\ |q_{y2} - o_{y2}| \\ \vdots \\ |q_{xn} - o_{xn}| \\ |q_{yn} - o_{yn}| \end{pmatrix}$$

where (q_{xi}, q_{yi}) and (o_{xi}, o_{yi}) are respectively the coordinates of the the projected and observed locations for the i^{th} feature. We aim to find a pose that results in projected feature locations that minimize this error. There is a chance that the estimated pose will satisfactorily minimize this error. If so then no further refinement of the pose is necessary. However, it is unlikely that our estimated pose will be satisfactory, in which case it is necessary to refine the pose as described below.

Iteratively refine pose estimate

The pose estimate is iteratively refined in the tracking loop in Figure 6.4. This loop consists of a five step process:

1. determine the Jacobian,
2. use this to calculate a pose correction factor with which to adjust the current pose estimate,
3. apply the adjusted pose to the 3D model,
4. determine the new projected feature points,
5. check the error between the projected and observed feature points, and repeat this process again if the error is not small enough.

Steps 3, 4 and 5 are identical to the *Apply pose to model*, *Determine projected feature locations*, and *Check if error is acceptable* phases described above for the earlier part of the algorithm. However, calculating the Jacobian and determining the pose correction factor are specific to the tracking loop and are discussed in detail here.

The pose correction factor is the central part of the tracking process. It enables the pose to be effectively adjusted so that it converges to a suitable value, minimizing the error between the projected and observed feature points. The Jacobian is simply a necessary prerequisite to calculating the pose correction factor.

The Jacobian \mathbf{J} is the matrix of partial derivatives of the projected feature locations with respect to the pose. Concatenating the (x, y) coordinates of the projected feature locations in a vector

$$\mathbf{q} = (q_{x_1}, q_{y_1}, q_{x_2}, q_{y_2}, \dots, q_{x_n}, q_{y_n})^T$$

the Jacobian is then calculated as follows

$$\mathbf{J} = \frac{d\mathbf{q}(\mathbf{p})}{d\mathbf{p}} = \begin{pmatrix} \frac{\partial q_1}{\partial p_1} & \frac{\partial q_1}{\partial p_2} & \dots & \frac{\partial q_1}{\partial p_6} \\ \frac{\partial q_2}{\partial p_1} & \frac{\partial q_2}{\partial p_2} & \dots & \frac{\partial q_2}{\partial p_6} \\ \vdots & \vdots & & \vdots \\ \frac{\partial q_{2n}}{\partial p_1} & \frac{\partial q_{2n}}{\partial p_2} & \dots & \frac{\partial q_{2n}}{\partial p_6} \end{pmatrix} \quad (6.2)$$

where p_i and q_i denote the i^{th} elements of \mathbf{p} and \mathbf{q} respectively,

$$\frac{\partial q_{i \in \text{odd}}}{\partial p_j} = \frac{-f}{z_i} \left(\frac{\partial x_i}{\partial p_j} - \frac{x_i}{z_i} \frac{\partial z_i}{\partial p_j} \right)$$

$$\frac{\partial q_{i \in \text{even}}}{\partial p_j} = \frac{-f}{z_i} \left(\frac{\partial y_i}{\partial p_j} - \frac{y_i}{z_i} \frac{\partial z_i}{\partial p_j} \right)$$

with f being the camera focal length, and x_i, y_i and z_i , the 3D locations of the i^{th} feature, given by

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \mathbf{R}(\mathbf{p})\mathbf{m}_i + \mathbf{t}(\mathbf{p})$$

$\mathbf{R}(\mathbf{p})$ and $\mathbf{t}(\mathbf{p})$ are the rotation matrix and translation vector defined by the current pose \mathbf{p} , and \mathbf{m}_i is the model coordinates of the i^{th} feature.

Once the Jacobian has been determined we are now able to calculate the pose correction factor \mathbf{c} using Lowe's algorithm (Lowe, 1991) as follows

$$\mathbf{c} = (\mathbf{J}^T \mathbf{J} + \lambda \mathbf{W}^T \mathbf{W})^{-1} (\mathbf{J}^T \mathbf{e} + \lambda \mathbf{W}^T \mathbf{W} \mathbf{s}),$$

where

$$\mathbf{W} = \begin{pmatrix} \frac{1}{\sigma_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2} & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & \frac{1}{\sigma_6} \end{pmatrix}$$

is a diagonal matrix whose diagonal elements are the reciprocals of the standard deviation of the change in parameter \mathbf{p}_i from one frame to the next, \mathbf{J} is the Jacobian matrix of \mathbf{q} defined in Equation 6.2, \mathbf{e} is the error vector containing the difference between the observed and projected 2D feature locations, s_i is the desired default value for parameter \mathbf{p}_i , and λ is a scalar weight.

By iteratively applying this correction factor a least squares solution can be achieved. This is an extension of Newton's method, designed to obtain stable approximate solutions in the presence of noise. The stabilisation technique uses the addition of a small constant to the diagonal elements of $\mathbf{J}^T \mathbf{J}$ in order to avoid the possibility of this matrix becoming close to singular.

In this algorithm, the standard deviation of parameter changes in consecutive frames represents the limit on the acceleration of each parameter from one frame

to the next. For translation parameters, a limit of up to 10 pixels (within the image size of 384×284) is used as the standard deviation, and for rotational parameters 0.1 radians is used. The scalar λ can be increased to strengthen the weight of stabilisation whenever divergence occurs. However, a constant scalar of 0.64 is used in this system as this was found to maintain stability throughout the iterations.

6.2.2 Mouth Detection and Correction for Pose

Template matching is used to track the corners, top and bottom of the mouth. The 3D pose determined from the head tracker is then used to determine the mouth feature locations in 3D from which the true width and height of the mouth can be determined.

Mouth Template Matching

The mouth corners are tracked using adaptable templates with the template search areas centred at the feature locations determined in the previous frame.

Once the mouth corners are located, the upper mouth edge is searched for along the line joining the mid-point of the mouth corners to the nose template position, and the bottom mouth edge is searched for on the line perpendicularly bisecting the line joining the mouth corners (see Figure 6.8). Locating the mouth edges on these lines avoids the potential problem of the templates drifting along the top and bottom mouth edges, and the computational requirement for template matching is drastically reduced by only searching along a line rather than a 2D region. As the head turns to the side and tilts the mid-point of the line joining the mouth corners no longer corresponds to the centre of the mouth, however, this approximation proved adequate for estimating the height of the mouth.

The adaptable templates are updated every frame. This updating is essential as the mouth corners look very different when the mouth is open as opposed to when it is closed.

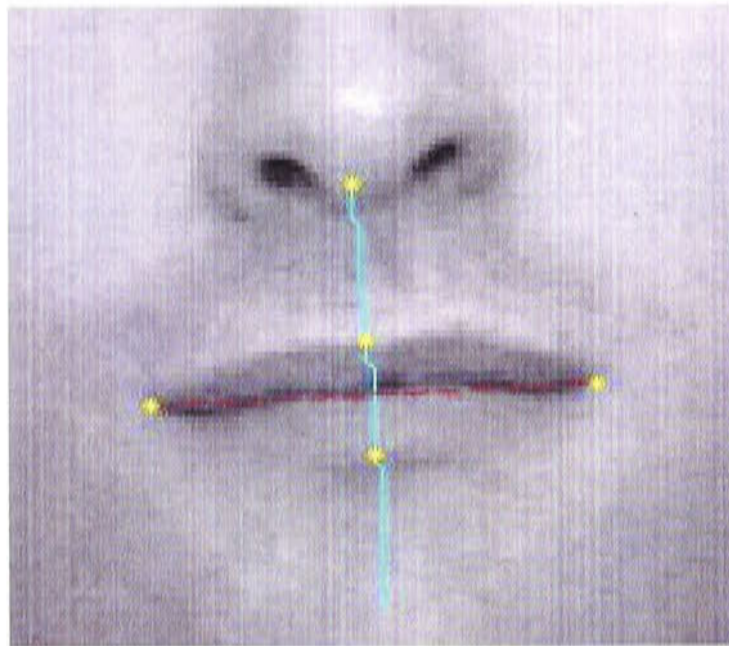


Figure 6.8: Search lines for locating the top and bottom mouth edges.

Determining Mouth Width and Height

Since the mouth is constantly changing shape during speech it cannot be modelled as a rigid 3D object. Instead the mouth is assumed to lie flat on a plane parallel to the front of the face. The observed 2D mouth feature locations are projected from the image plane onto this face plane as illustrated in Figure 6.9, the orientation of the face plane is given by the head pose obtained from the 3D head tracker. Thus the 3D locations of the mouth features \mathbf{b}_i are determined.

The mouth height and width are then calculated as the 3D Euclidean distances between the top and bottom mouth edges and the mouth corners respectively.

$$\text{width} = \|\mathbf{b}_1 - \mathbf{b}_2\|$$

$$\text{height} = \|\mathbf{b}_3 - \mathbf{b}_4\|$$

6.2.3 Experimentation

The system was implemented in Matlab 5.3 on a standard 600 MHz Pentium III, and tested on a 234 frame of video sequence recorded off-line at 25 Hz. Each frame was 8-bit grey-scale and 384×284 pixels. Feature templates were chosen to be 12×12 pixels, and the search region size was set to 36×36 .

Three non-deformable features were used to track the head: the eye corners and the edge of one nostril. More features can be used, but three is sufficient to

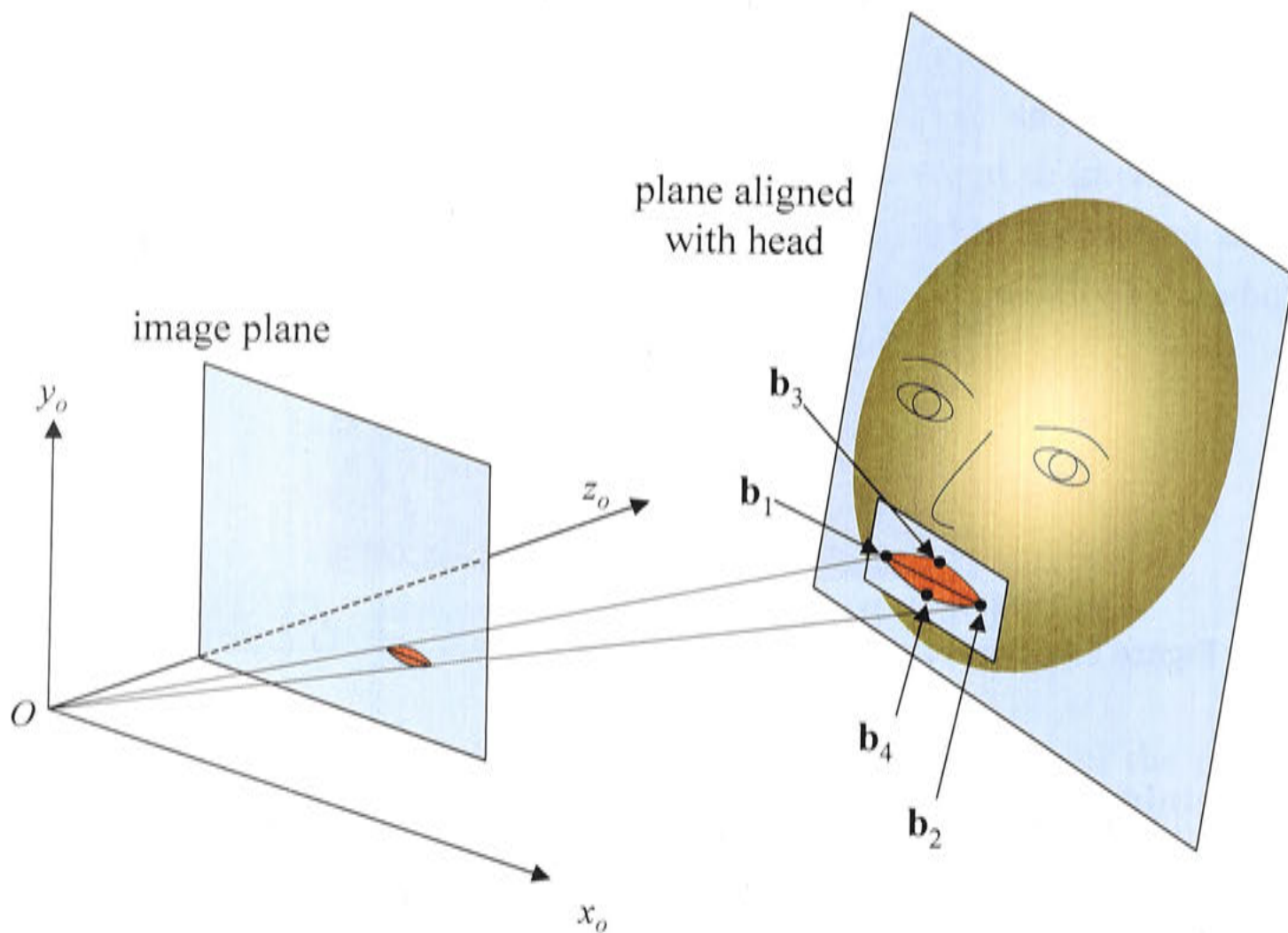


Figure 6.9: Projection of mouth points from image plane to face plane.

determine the 3D pose of the head. The mouth corners were also tracked, but these were not used to estimate the head pose and were instead used exclusively for determining the mouth shape.

In order to verify the suitability of the algorithm for realtime applications the Matlab flop counter was used to estimate the average number of floating point operations required per frame. Table 6.1 shows the average number of floating point operations per frame for different steps of the algorithm calculated over a 234 frame test sequence. The vast majority of computations are taken by the template matching, which involves calculating the normalised cross correlation of the templates at each location across the search regions. Using the initial sparse search, as detailed in Section 6.2.1, has made this computation considerably less than the 4.9 Mflops it would otherwise be. It can be reduced further to 0.89 Mflops by increasing the sparsity from two to three and performing the sparse search at every third (rather than every second) row and column. However, increasing the sparsity of the search increases the likelihood of the tracker losing the features. In any case, with a sparsity of two the total number of computations per frame is within an amount that is feasible to perform in realtime were the algorithm to be implemented in C/C++.

Table 6.1: Estimated Computations per Frame

Stage	Mean computation per frame (flops)
Pose estimation	1,270
Template matching	1.54×10^6
Object tracking	13,400
Correcting mouth shape	705

Figure 6.10 shows six snap shots of the system whilst tracking, the animated face illustrates the current pose and mouth dimensions of the subject. The full tracking sequence is included on the CD-ROM enclosed with this thesis.

The quantitative output of the system over the full video sequence is presented in Figure 6.11. This shows a record over the test sequence of the 3D head pose, the uncorrected mouth dimensions and the corrected mouth dimensions. As expected there is a strong correlation between the amount the mouth width is corrected and the head's rotation about the vertical y -axis.

The result shows the 3D head tracker recovering the head pose, and the mouth shapes being effectively corrected throughout the sequence. For example, at the frames 40 and 95, the phonemes "W" and "long-E" were spoken with the speaker's head nodding. Thus the detected mouth heights for their surrounding frames were corrected as shown in Figure 6.11(c). In frames 95 and 176 (phonemes "long-E" and "M" respectively) the speaker's head was turned to the side and it was necessary to correct the mouth width. Enunciating these phonemes causes the mouth to widen relative to the neutral mouth width. Figure 6.11(d) shows that the uncorrected mouth widths for these frames were similar to the neutral mouth width, and that the correction is successfully made to enlarge the widths to represent the actual mouth shape corresponding to these phonemes.

These results demonstrate the usefulness of this method for correcting the observed size of the mouth as a speaker moves his head in 3D. Mouth height and width are the two primary visual quantities used by audiovisual speech processing systems. Being able to estimate these correctly whilst accommodating for head translation and rotation is essential for audiovisual speech recognition in a real world environment where speakers naturally move their heads around during speech.

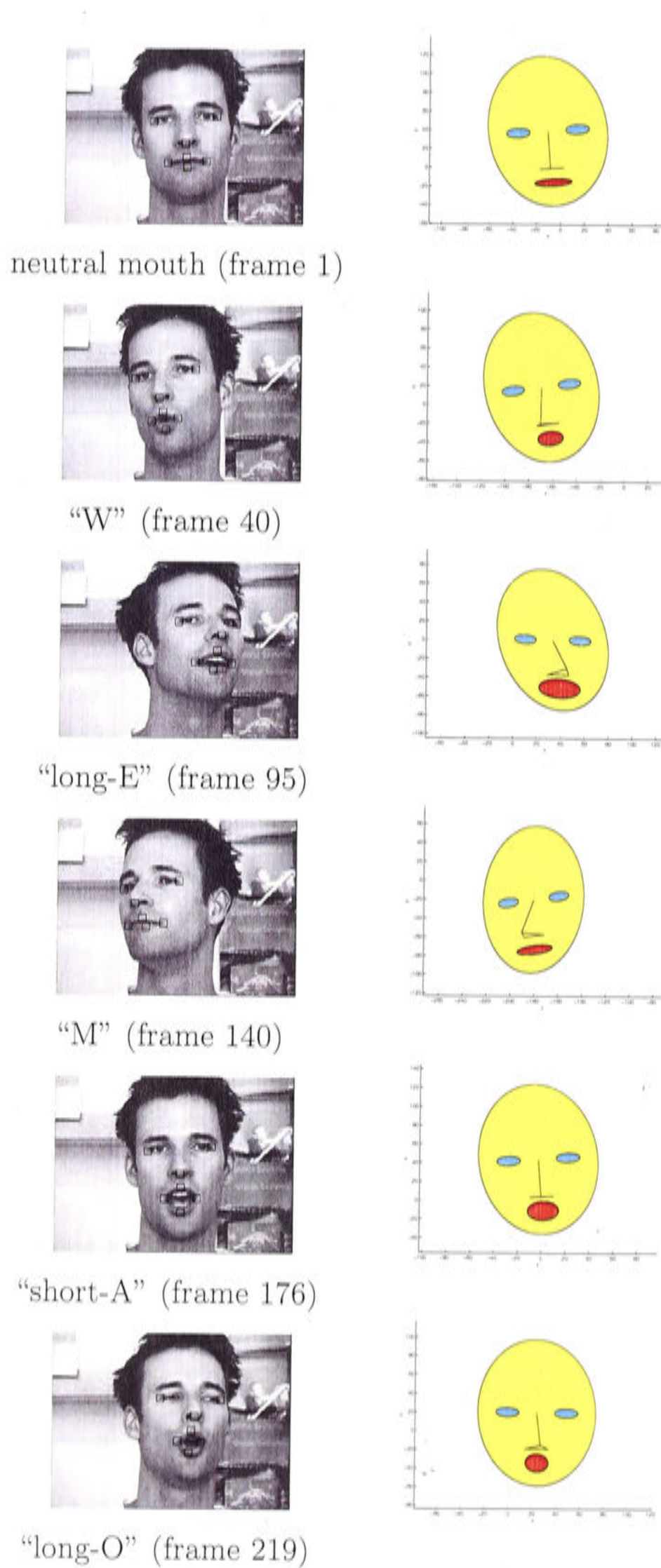


Figure 6.10: Some snapshots of the system in operation. The phoneme being pronounced and the frame number are indicated.

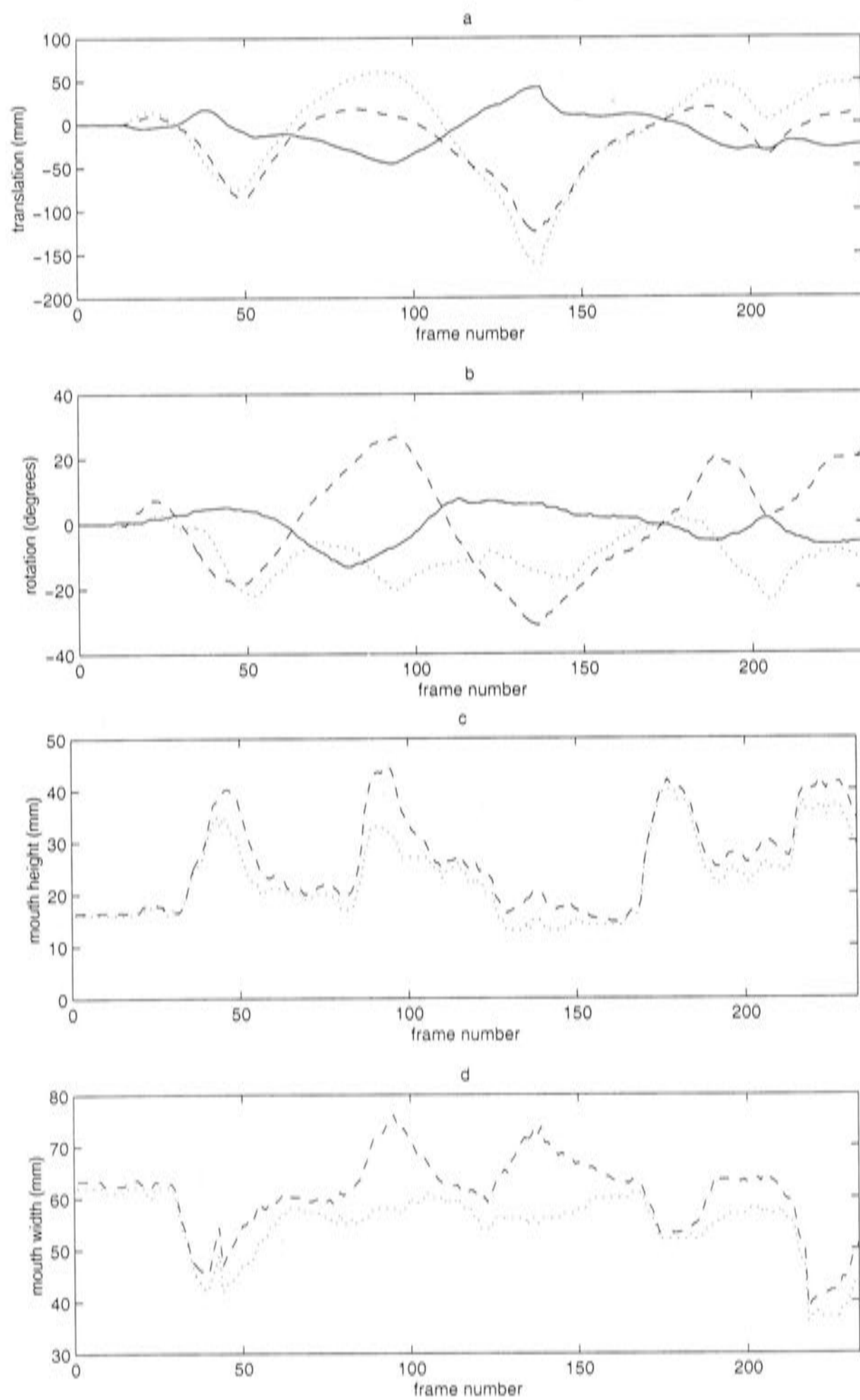


Figure 6.11: Results of the head and mouth tracking system. (a) Head translation, x , y and z translations are shown dotted, dashed and solid respectively. (b) Head rotation, rotations about x , y and z axis are shown dotted, dashed and solid respectively. (c) and (d) Mouth height and width, corrected dashed and uncorrected dotted.

6.2.4 Section Review

A system has been developed that measures the dimensions of a speaker's mouth whilst the speaker's head is moving and exhibiting rotations of up to 30 degrees away from the camera. Our system tracks the pose of the speaker's head in 3D, detects the unadorned mouth by tracking the corners and a point on the upper and lower edge of the lips, and estimates the mouth height and width during speech. The system is demonstrated on a person speaking whilst moving his head in 3D, and the mouth height and width are corrected over 9 seconds of 25 Hz video footage.

This section has demonstrated an application of measuring the mouth height and width on a moving head using typically noisy pose information from a monocular head tracking system. In the next case study a stereo system is presented that uses accurate head pose data from an established stereo tracking system and extends the mouth tracking to recover the full 3D shape of the outer lip contour.

6.3 Stereo Lip Tracking

This section describes a stereo lip tracking system that tracks a person's unadorned lips in 3D, and outputs the 3D locations of the mouth corners and a set of points around the outer lip contour. This output is suitable for audio visual speech processing, 3D animation, or expression recognition. A stereo head tracker is used to track the subject's head, allowing for robust performance whilst the subject's head is moving and turning with respect to the cameras. The head pose is used in conjunction with the novel *adaptable templates* described in Section 6.1 to robustly estimate the locations of the corners of a deforming mouth. A 3D geometric model is used to generate search paths for key points on the outer lip contour, and these are subsequently located using adaptable templates and stereo matching. The system is demonstrated robustly tracking the head pose and 3D mouth shape of a person speaking while moving his head.

Figure 6.12 shows the key components of the system. The main focus of this section will be on the lip tracking component, but first Section 6.3.1 describes the stereo vision system used to capture images of the subject, and Section 6.3.2 briefly summarises the head tracker. Section 6.3.3 then covers the lip tracking system in detail, Section 6.3.4 presents some results of the system in operation,

and Section 6.3.5 concludes with a review and some suggestions for future work.

6.3.1 Stereo Vision System

The configuration of the stereo vision system is shown in Figure 6.13. The two cameras are positioned equidistant from the origin and are verged (angled towards the origin in the horizontal plane) at about 5 degrees. This is designed to offer the best measurements of an object the size of a human head placed approximately 600mm in front of the cameras, and provides an effective working volume 20 cm high \times 30 cm wide \times 50 cm deep for 3D tracking.

Both cameras are standard, colour analog NTSC video cameras whose outputs are multiplexed into a single channel before being acquired by a Hitachi IP5005 video card. The result is a 512×480 colour image, captured every 33ms, where the top half contains the right hand image and lower half the left hand image.

6.3.2 Head Tracking

Our system uses the stereo head tracker presented by Matsumoto and Zelinsky (2000) and detailed in Chapter 2. This is an early prototype of the Seeing Machines FaceLab system. It was developed in our lab and is the combined work of Zelinsky, Heinzmann, Matsumoto, Newman and Rougeaux. The system uses calibrated stereo cameras and returns accurate estimates of the head pose with position measurement within ± 1 mm and orientation ± 2 degrees. The system requires no markers or special make-up to be worn, and runs in realtime on a standard PC.

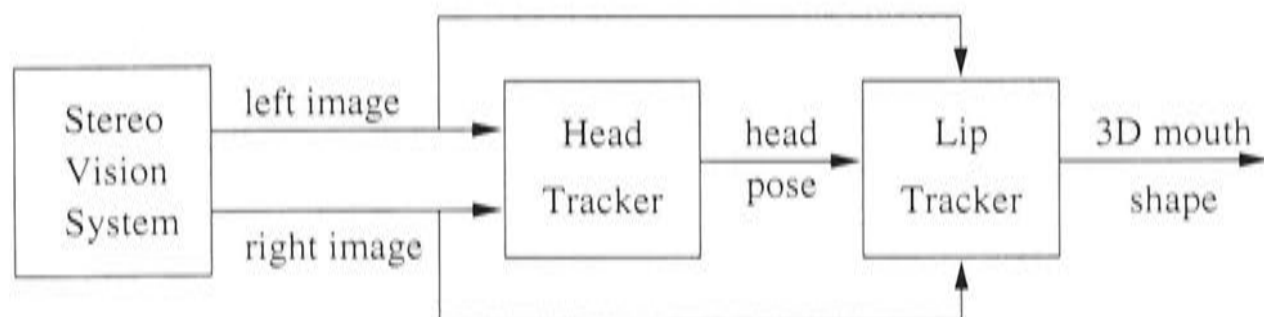


Figure 6.12: Overview of the system.

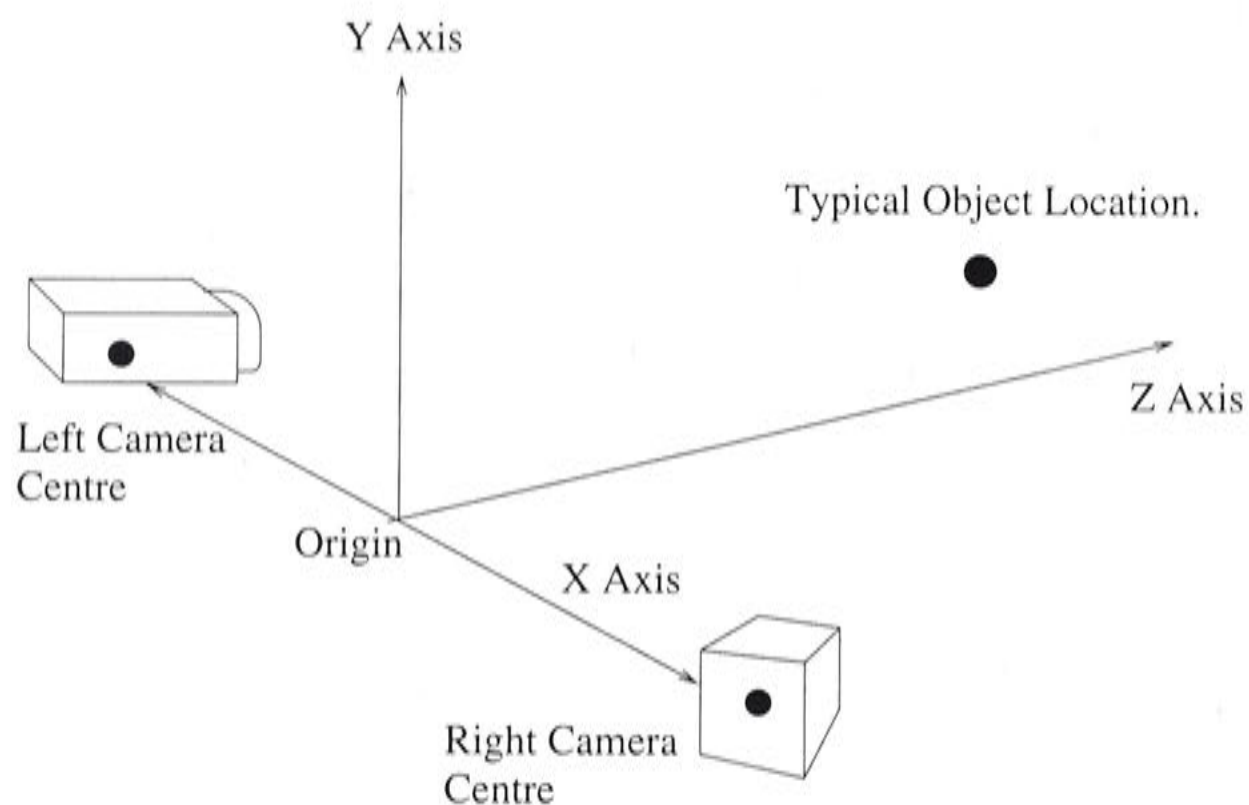


Figure 6.13: The stereo camera arrangement.

6.3.3 Lip Tracking

The mouth is a deformable feature, however, like the eyebrows, eyelids and other deformable facial features, it is firmly attached to the head and is only able to deform in an elastic manner. Our lip tracking system uses head pose information from a head tracker to localise the approximate location of the mouth in each image. Adaptable templates are used to track individual mouth features and accommodate for the elastic deformation of these features as the mouth shape changes, and stereo matching these features enables us to determine the 3D shape of the deforming mouth.

We define three primary tracking points, the left and right mouth corners and the outer edge of the centre of the upper lip, as illustrated in Figure 6.14. These primary features are chosen owing to the importance of their locations when determining the lip contour, and their distinctive appearance that allows them to be more easily tracked than other points on the lip contour. The locations of the primary features will later form the foundation of our search procedure for tracking the lip contour.

Before commencing tracking it is necessary to initialise the system. This is currently performed manually and is done on a frame where the subject is facing close to front-on to the cameras and the mouth is in a neutral (closed and re-

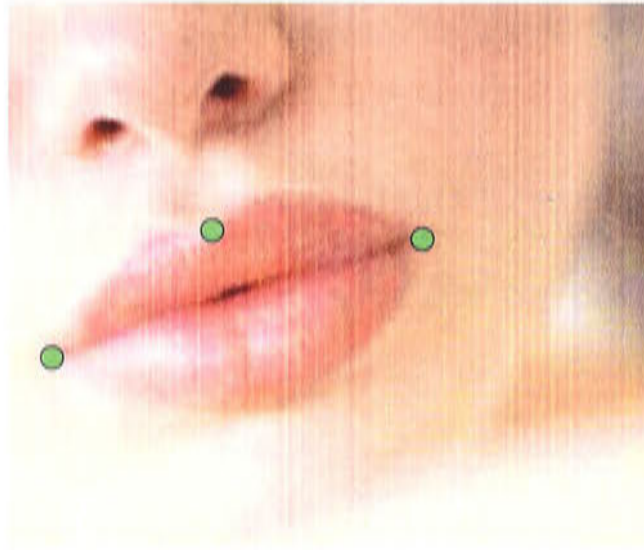


Figure 6.14: The lip corners and the centre of the upper lip contour provide the primary tracking points for our system.

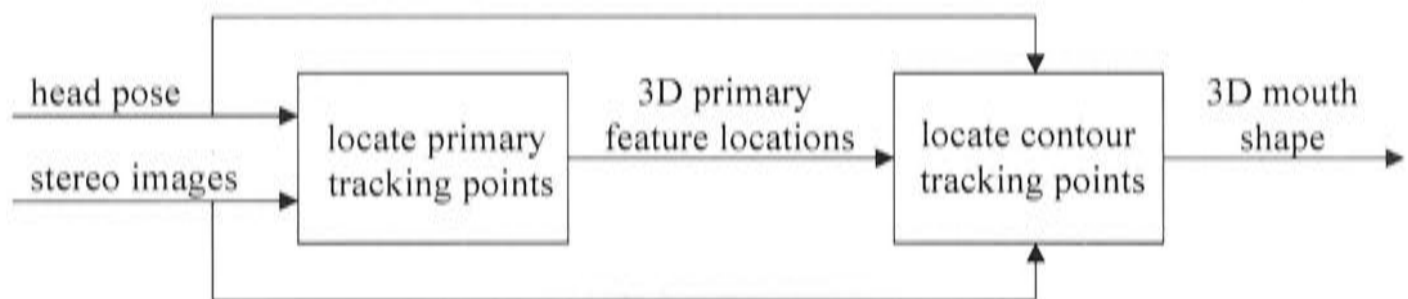


Figure 6.15: Lip tracking system.

laxed) position. The mouth corners and centre of the upper outer lip contour are identified, and these are matched in the other stereo image to determine the 3D locations of these points on the neutral mouth. In addition to this, the initial locations of a set of tracking points on the outer lip contour are manually identified. These are evenly spaced from left to right across the upper and lower lips so they lie on the lip contour search lines that will be described below in Section 6.3.3.

Figure 6.15 summarises our approach to detecting the lip contour. Firstly we establish the 3D locations of the primary tracking points, these define search lines for the contour features, and then the locations of the contour features are identified.

Search Structure

As the mouth deforms during speech and through facial expressions its appearance and shape can change drastically. In order to effectively track the mouth contour we adopt a search structure that allows us to reliably locate specific

mouth features. This approach is a further development of our previous work in 2D lip tracking presented in Section 6.2.

The search procedure consists of the following steps, which shall be discussed in more detail hereafter.

1. Locate primary features:

- (a) the head pose is used to determine the 3D locations of the primary features if they were in the neutral position as they were at initialisation,
- (b) search areas for the primary features are defined from the projected positions of these 3D feature locations,
- (c) the current primary feature locations are identified, and
- (d) the 3D location of the primary feature is determined via stereo matching.

2. Locate lip contour tracking points:

- (a) based on the 3D locations of the primary features, search lines are defined for locating the outer lip contour,
- (b) the outer lip contour is identified, and
- (c) the 3D locations of the outer lip contour tracking points are determined via stereo matching.

Locating Primary Tracking Points

The corners of the mouth are the most suitable points for tracking due to their distinctive (albeit drastically changeable) appearance. Their location is constrained to a small region of the face. Each mouth corner will always be in a region centred at the neutral mouth corner location. Because this region is quite small (typically $50 \times 50\text{mm}$) it is feasible to search the whole region in every frame. The other primary tracking feature, the centre of the upper lip, has the advantage that it does not move or deform as much as the mouth corners. However, its appearance is less distinctive as it can look quite similar to other points on the mouth contour, so it is typically not tracked as accurately as the mouth corners.

It is feasible to avoid tracking the centre of the upper lip and simply use the estimated location of the neutral mouth centre to locate the central primary feature. However, despite the lack of gross movement typically displayed by this feature, it is preferable to explicitly track the feature to accommodate for any movement that does occur. Nonetheless for the purpose of defining search lines for the lip contour, which will be discussed below, estimating the upper lip centre in the first image is still satisfactory.

In the initialisation process we determined the 3D locations of the primary tracking points in the initial frame where the mouth was in a neutral position. We now estimate the 3D locations of these points under the current head pose, then project them into the current image using pinhole camera projection, to estimate the approximate location of the neutral mouth points. The search for the current locations of the primary features can then be limited to 40×40 pixel regions around the estimated location of the appropriate neutral mouth feature.

The primary features are tracked within these search regions using the same method applied for the 2D lip tracker presented in Section 6.2. That is, normalised cross correlation with adaptable templates, using the sparse search approach described in Section 6.2.1 to reduce the computation.

Once the features have been located in the first stereo image, a larger template is taken from around each tracking point in this image and used to locate the feature in the second stereo image using normalised cross correlation. Since the stereo system is calibrated it is only necessary to search along the epipolar line in the second image. Once this is done the 3D location of the primary tracking points are determined via linear triangulation (see Section 2.4.1).

Locating Lip Contour Tracking Points

The primary tracking points define the outer edges of the mouth and the center of the upper lip. However, to fully describe the mouth shape it is necessary to track points all along the mouth contour. Our system uses a set of tracking points to characterise the outer lip contours of the upper and lower lips. The number of points used is at the discretion of the user, with the limiting factor being the additional computation required for more points. In our experiments we used 18 contour tracking points which provided a detailed estimate of the lip shape.

The contour tracking points are located on search lines parallel to the vertical

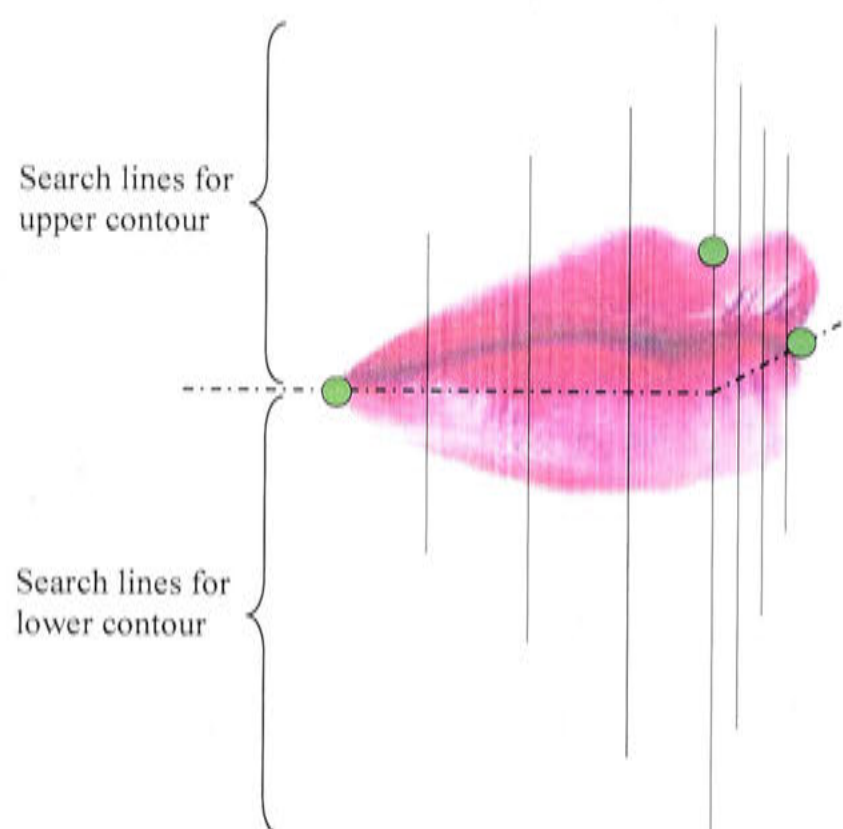


Figure 6.16: Search lines are placed in 3D aligned with the vertical head axis and equidistant between the primary tracking points, shown in green.

head axis. The search lines are initially defined in 3D, and are automatically located based on the locations of the mouth corners and the centre of the neutral mouth. Search lines are placed on each side of the mouth, as shown in Figure 6.16 spaced equidistant between the mouth corner and the centre of the neutral mouth, if an odd number of points is used to track the upper and lower lips then a search line is placed at the centre of the mouth as well. These 3D lines are projected onto one of the stereo images to provide the set of 2D search lines on which to locate the outer contour in the image.

For the upper lip contour the search lines start at the level of the lip corners and extend upwards (in the positive y direction in the head reference frame, see Figure 6.3). For the lower lip contour the search lines start at the level of the lip corners and extend downwards in the opposite direction. The length of each line is proportional to how central the point is, and is chosen to generously accommodate the full range of mouth motion (the length is defined in 3D, before the lines are projected onto the image plane).

Locating the mouth edges on these lines avoids the potential problem of the templates drifting along the top and bottom mouth edges, and the computational requirement for searching is drastically reduced by only examining a line of points,

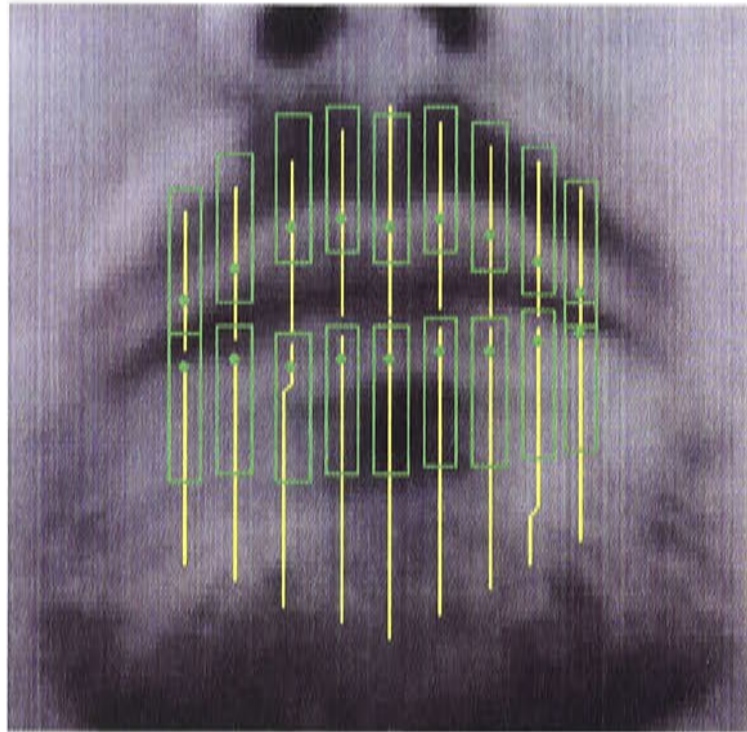


Figure 6.17: Contour tracking templates. Feature locations and template boundaries are indicated in green, search lines are shown in yellow.

rather than a 2D region of the image.

As the mouth opens and closes the appearance of the interior of the mouth changes drastically, due not only to the changing shape of the mouth, but the appearance of the teeth, tongue and oral cavity. On the other hand, the facial region outside the mouth changes much less. For this reason we chose to track the mouth contour with elongated offset templates stretching away from the mouth as shown in Figure 6.17. By including a large proportion of the external (more constant) region surrounding the mouth, in addition to the lip, and *not* including the inner mouth region, these templates can detect the outer lip contour whilst remaining robust to changes in the appearance of the inner mouth region.

Normalised cross correlation with adaptable templates is used to locate the contour tracking points on the search lines in one of the stereo images. Once the features have been located, wider templates are taken from the current image around each tracking point and used to locate the feature in the second stereo image. The 3D search lines are also projected onto the second image, to determine the approximate location of the features. Since the stereo system is calibrated it is only necessary to search along the epipolar line in the second image, and the range of disparities is reduced by only searching close to the intersection of the epipolar line and the appropriate search line.

6.3.4 Experimentation

A significant amount of experimentation has been carried out to analyse the performance of the head tracker using a mannequin head mounted on a pan-tilt device (Newman *et al.*, 2000). The head tracker has been shown to accurately recover the head pose position within 1 mm and pose angle to within 2 degrees. It can accommodate head velocities of up to 100 degrees per second and head rotation up to 45 degrees away from the cameras.

The lip tracking system was implemented in Matlab 5.3 on a standard 600 MHz Pentium III, and tested off-line on a 148 frame video sequence recorded at 30 Hz. Each frame was 8-bit grey-scale and 240×320 pixels, and was accompanied with head pose data output from the face tracker. The primary features were initially located with 10×10 templates, then 20×20 templates were used to perform the stereo matching. The elongated offset templates used to track the lip contour were chosen to be 20×5 and wider 20×30 templates were used to determine the stereo correspondence in the second image.

The lip tracker was shown to track the mouth throughout a sequence of footage of a subject moving his head and mouth in 3D. Figure 6.18 shows several snapshots of the system in action. Both left and right stereo images are shown, the primary tracking points are indicated in blue and the contour tracking points are shown in red. The 3D mouth shape is also shown. The full video sequence of the tracker in action is included on the CD-ROM enclose with this thesis.

The results presented here have not been smoothed or filtered at all and represent the raw data output from the feature tracking technique described. For animation purposes or to make a smoother moving reconstruction the 3D locations could be filtered to remove high frequency motion.

Quantitatively verifying the performance of the lip tracker required manually locating tracking points throughout a sequence and comparing these with the automatically tracked points generated by the tracking system. This was done for the three primary tracking points (the mouth corners and the centre of the upper lip) over the 148 frame sequence. The absolute 3D error was recorded along with the absolute error in the x , y and z directions², and the results are presented in Figures 6.19 to 6.22. The mean absolute errors are shown in Table 6.2.

²These Cartesian directions refer to the axes in the world coordinate frame located between the two cameras as shown in Figure 6.13.

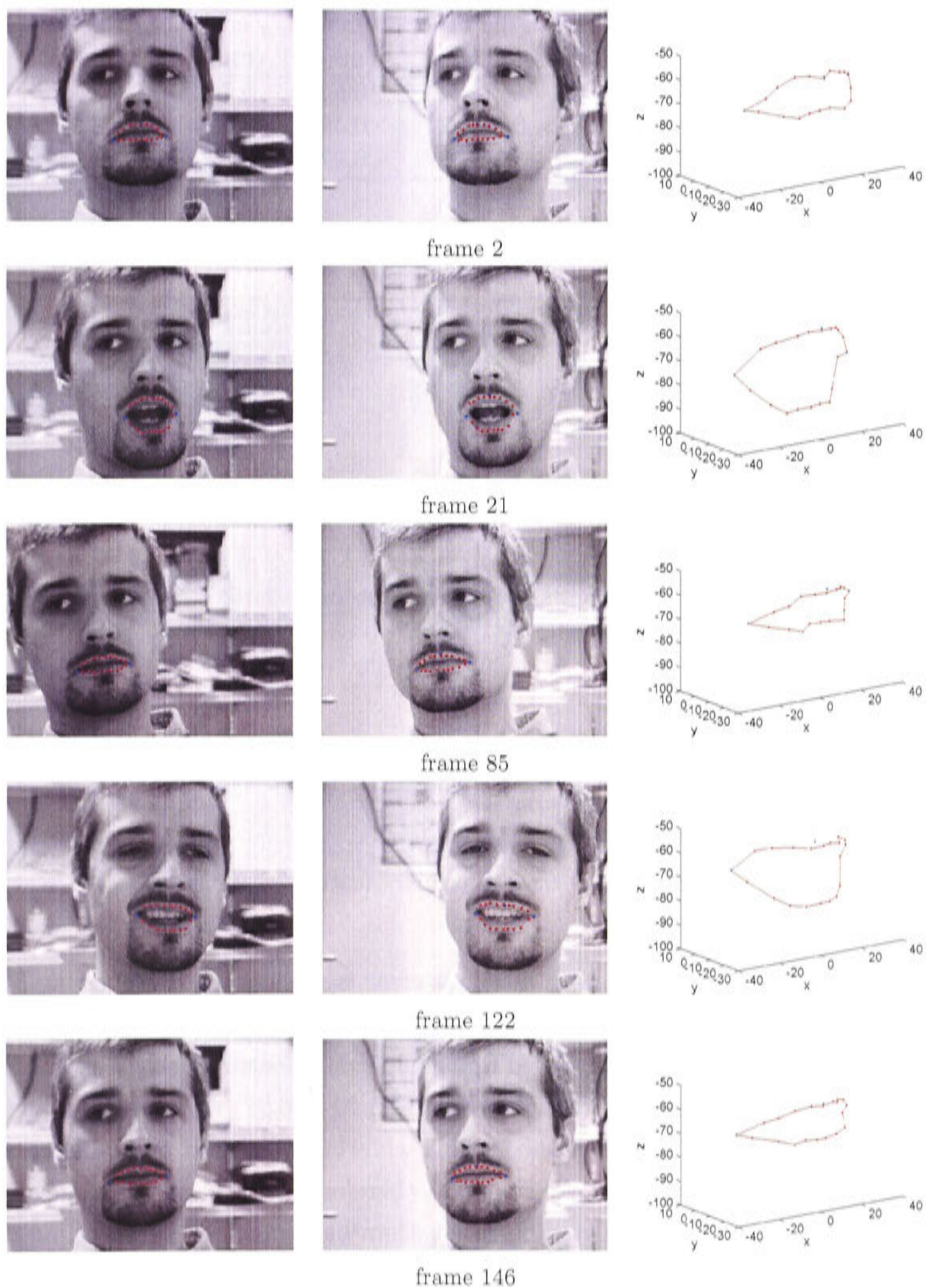


Figure 6.18: The system in operation. The first two columns show the left and right stereo images and lip tracking points, the third column shows the 3D mouth shape in the head reference frame (dimensions in mm).

Table 6.2: Mean absolute error in primary tracking points.

Feature	Mean 3D error (mm)	x error (mm)	y error (mm)	z error (mm)
right mouth corner	1.2	0.53	0.76	0.46
lip centre	0.69	0.40	0.22	0.35
right mouth corner	0.7	0.47	0.32	0.21

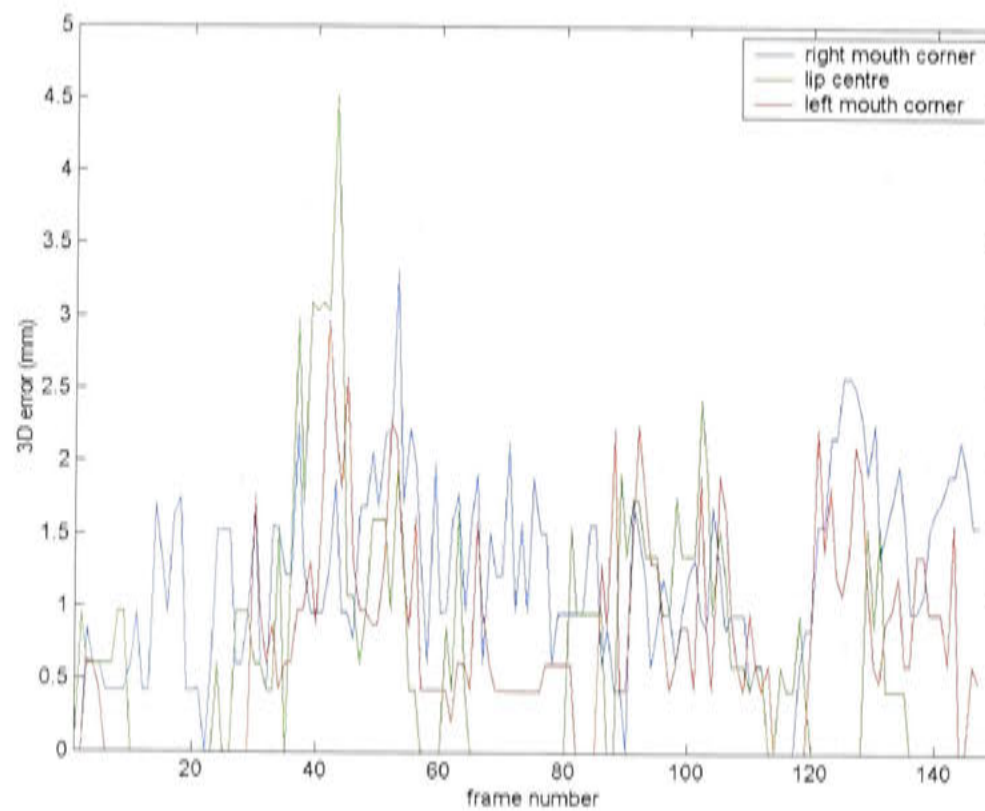


Figure 6.19: 3D error in primary feature locations.

From Table 6.2 we see the mean absolute error is negligible. However, as Figure 6.19 shows there are isolated errors in the tracked feature positions of up to 4.5mm. The largest error is observed for the lip centre feature, and by examining the errors in the x , y and z directions (Figures 6.20 to 6.22) we see this is caused by a large error in the z -depth of the feature, while errors in the x and y -directions are insignificant at this point. We conclude, therefore, that this single large error was caused by poor stereo matching rather than bad tracking. However, on the whole the stereo matching performs well, as can be seen from Figure 6.22 that shows that the z -error seldom rises above 1.5 mm.

Errors in the x and y directions shown in Figures 6.20 and 6.21 are indicative of how well the adaptive templates are tracking. The results are encouraging with the error seldom rising above 2mm, and averaging well below 1mm for each feature (see Table 6.2).

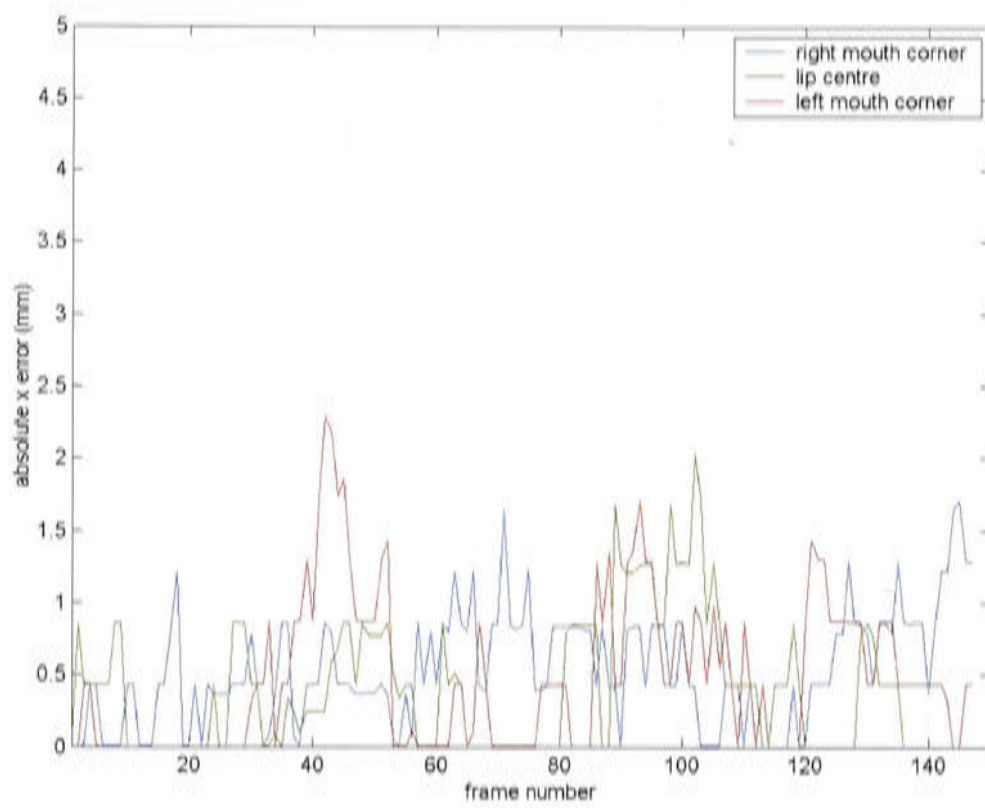


Figure 6.20: Absolute error in x -direction in primary feature locations.

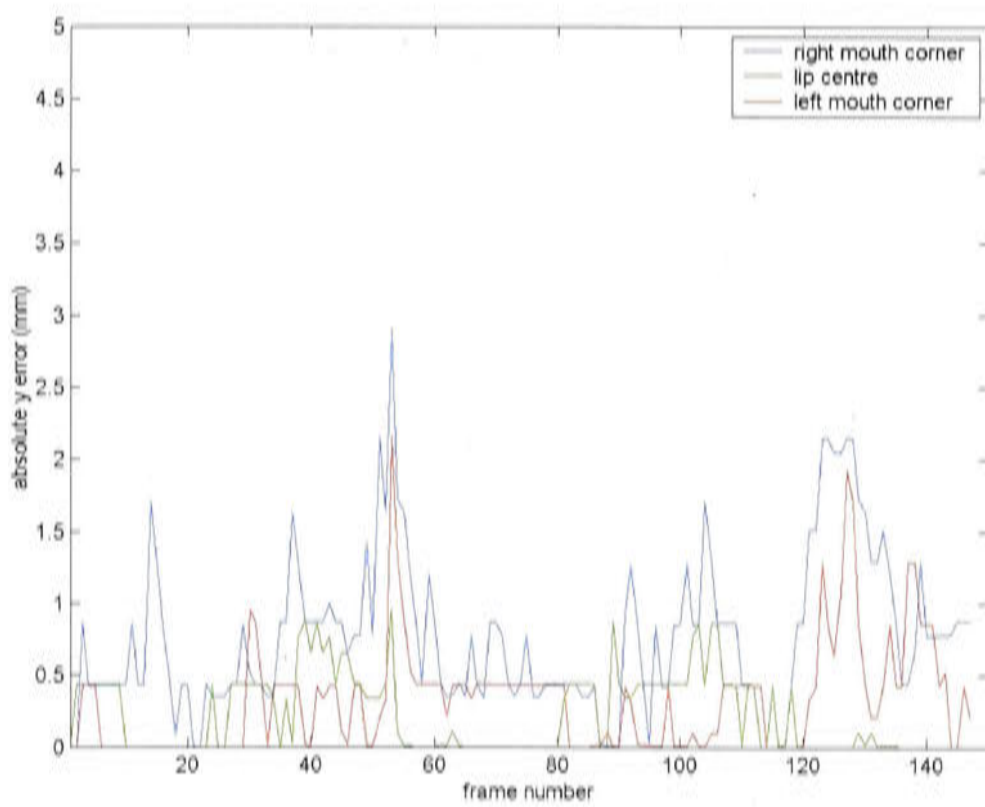


Figure 6.21: Absolute error in y -direction in primary feature locations.

To verify the suitability of the algorithm for realtime implementation estimates of the average number of floating point operations required per frame were determined using the Matlab flop counter. Table 6.3 shows the average number of floating point operations per frame for different stages of the algorithm, these figures were calculated over a test sequence of 148 frames.

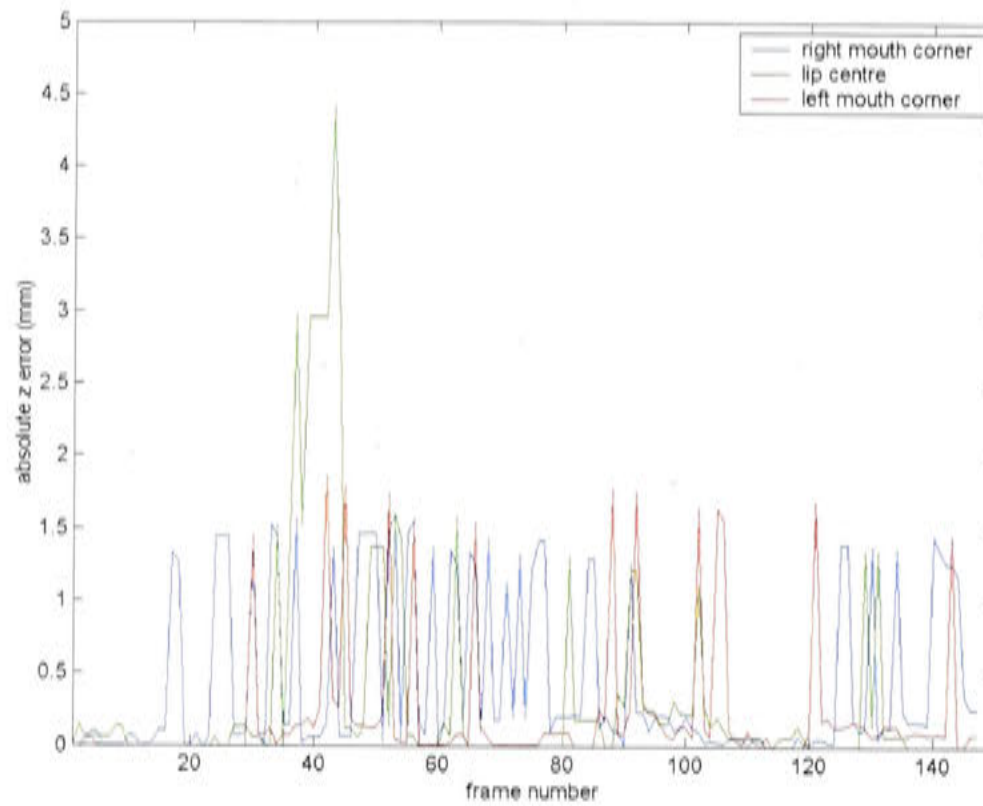


Figure 6.22: Absolute error in z -direction in primary feature locations.

Table 6.3: Estimated Computations per Frame

Stage	Mean computation per frame (Mflops)
Estimate primary feature locations	0.000264
Locate primary features	6.47
Define search lines	0.00772
Locate 18 contour points	1.63
Overall	8.26

The majority of time was spent locating the three primary feature points due to the heavy computational load of calculating the normalised cross correlation over the 2D search regions. The computations required to locate all 18 contour tracking points is much less, although there are six times as many points, this is because these were located on lines (rather than within 2D regions) so the search for each contour point only needed to be carried out in 1D. The computational requirement for the rest of the algorithm is minimal. Since we do not fit a model to our data the bulk of our effort is expended in locating the tracking points. Estimating the initial locations of the primary features and defining the contour search lines use a trivial amount of computation compared to that required for the correlation used to locate the features. The computation results here are for the lip tracker alone and do not include the operation of the head tracker.

6.3.5 Section Review

This section has presented a technique to track the 3D shape of a deforming mouth whilst the subject's head is moving in 3D. The mouth corners are tracked along with numerous points around the outer lip contour, and the 3D locations of each of these points determined via stereo correspondence. The lip tracking results presented were generated off-line, although the technique is efficient enough for realtime implementation.

The lip tracker presented here is the first system to track the raw 3D mouth shape. Some previous methods (see Chapter 2) have inferred the 3D mouth shape from predefined mouth models, but none have used stereo to directly measure the 3D locations of tracking points.

Our technique is quite different from the active contour approach to lip tracking (Kaucic *et al.*, 1996, for example). Active contours use an underlying model to constrain the shape of the contour, and the visual sensing is just a single component of the tracking procedure. Our method is directly based on visual information and the shape of the mouth contour is much less constrained. The advantage of our approach is its simplicity and ability to track diversly shaped contours. The disadvantage is that it is not forced to be "mouth-like", and it relies entirely on the image information along its search lines to locate the contour correctly. This, however, is achievable with the use of adaptable templates. As an aside, the adaptable templates we use for tracking could be used to good effect to provide visual information in an active contour system.

At present the lip tracking system relies only on gray-scale intensity information. The system could be extended to utilise a number of other cues that could be merged together to increase the robustness and versatility of the system. Colour image information is one such cue which shows great promise for lip tracking and analysis of the mouth region (Goecke *et al.*, 2000). An area-based stereo depth map is another.

The template matching-based approach adopted in this chapter restricts the tracking to the outer lip contour. Whilst this is arguably the more important contour for animation and visualisation purposes, the inner contour is much more useful for audio-visual speech processing, since it is the inner contour which defines the airflow in and out of the mouth (Goecke *et al.*, 2000). Ultimately it is desired for a lip tracker to track both inner and outer contours in 3D. However,

the inner contour is an elusive target, and is difficult to define in 3D. Unlike the outer contour which is a distinctive line on a surface, the inner contour is the boundary between the lip and the oral cavity space and has no definite 3D location. Any attempt to determine the inner contour in 3D will benefit strongly from knowledge of the outer contour that is located by the system presented here.

6.4 Summary

This chapter has addressed the problem of tracking deformable facial features by examining two case studies of different lip tracking systems. The first used a monocular camera and was able to track the head pose and measure the mouth height and width of a subject. The second used a prototype of the commercially available FaceLab system to provide robust head pose information from stereo cameras, and then proceeded to track the 3D shape of the mouth as it deformed during speech. The concept of adaptable templates were also introduced, a technique designed for robust tracking of elastically deformable features, and these were used in both the lip tracking systems described to track points on the lip contour.

The stereo 3D lip tracker presented here is the first system to track the raw 3D mouth shape. As discussed in Chapter 2, some previous approaches have used complex models to infer the 3D mouth shape from monocular image information (Revéret and Benoît, 1998), but such inferences are restricted to the limited range of mouth shapes spanned by the models. Whilst these models are useful for speech processing, they are too constrained for character animation and other applications that require tracking the true, unconstrained 3D shape of the mouth.

Tracking deformable facial features is one of the ultimate end goals of computer vision systems aiming to enhance human computer interaction. After all, it is the actions and expressions of facial features that we humans use when interacting with someone and “reading” their face. While current systems are still very crude, the potential in this area is huge, and over the next decade we can expect to see computer vision systems capable of realtime lip reading, marker-less facial animation, and expression recognition.

Chapter 7

Conclusion

Enabling machines to see people is a crucial step towards machines that we can interact with as we do with other people. This thesis has worked towards a computer vision system that enables a computer to see people's faces, providing a basis for more natural and meaningful interaction between humans and machines.

We define what it means for a computer to "see" people as being able to locate and track humans in image sequences, preferably in realtime, and with robustness to different people's appearances and the operational environment. This task was divided into three subproblems:

1. **face localisation**, locating where a person is in a scene,
2. **face registration**, identifying facial features, and
3. **face tracking**, tracking the head pose and movement of facial features.

Novel solutions have been presented to each of these human tracking problems, that could potentially form an all-inclusive vision system allowing a computer or robot to see a person's face.

7.1 Summary

Chapter 1 opened with an introductory discussion to motivate and contextualise our research. The importance of visual information during interpersonal interaction and human-machine interaction was discussed, and it was concluded that

enabling a computer to see people is a significant step towards a computer that we can interact with like we do with other human beings.

In Chapter 2 we moved on to discuss related work, and reviewed the application of computer vision to locating and tracking people, in particular the face. The physical qualities governing the appearance of a face were examined, and visual cues suitable for detecting faces in images discussed. In addition a review of previous research was presented across the areas of face localisation, face registration, and face tracking.

Radial symmetry was found to be particularly useful for detecting facial features. However, the existing techniques with the most promising results suffered from high computational complexity and slow run-times making them inappropriate for realtime applications. In Chapter 3 we present a new image transform — the Fast Radial Symmetry Transform (FRST) — that allows efficient computation of radial symmetry in realtime. The FRST is a powerful visual cue for face detection and is used in several of the systems described in this thesis.

The first step to enabling a computer to see a person's face is to localise and track the approximate location of the face in an image sequence. New and innovative techniques are constantly being developed to track faces in images, however, despite the impressive results obtained, it is clear that no single cue can perform reliably in all situations. The key to an efficient and robust vision system for tracking faces or other targets is to intelligently combine information from a number of different cues, whilst effectively managing the available computational resources. In Chapter 4 we develop a system that does just this: adaptively allocating computational resources over multiple cues to robustly track a target in 3D.

After locating and tracking the location of a face in an image sequence, the next step towards enabling a computer to see a face is face registration, that is, detection of facial features and the verification of the presence of a face in an image. In Chapter 5 we present a case study of an automatic face registration system, designed to automatically initialise features for a head tracker. Once face registration is complete the computer has verified whether or not an image region contains a face, and if a face is present, the facial features are detected and ready to be tracked.

In Chapter 6 we explore the problem of tracking the face and deformable facial

features. This involves tracking both rigid and deformable facial features to fully characterise both the 3D head pose, and describe the locations of facial features relative to the head. We track the 3D pose of the head using predominantly rigid facial features, and then track the locations of deformable features relative to the head. A new form of templates is introduced to facilitate tracking deformable features, and these are used in two case studies. The first is a monocular lip tracker, and the second is a stereo lip tracking system that tracks the mouth shape in 3D.

7.2 Achievements

7.2.1 Fast Detection of Radial Symmetry

A new image transform, the Fast Radial Symmetry Transform (FRST), was presented. This transform utilizes local radial symmetry to highlight points of interest within a scene. With its low computational complexity and fast run-times it is well suited for realtime vision applications. Indeed, a realtime implementation of the transform was presented demonstrating its effectiveness as a cue for highlighting peoples eyes as they moved in front of the camera. The transform's performance was tested on a variety of images and compared with leading techniques from the literature. Both as a facial feature detector and as a generic region of interest detector, the FRST was seen to offer equal or superior performance to contemporary techniques at a relatively low computational cost, and provides a valuable cue for detecting eyes and other radially symmetric features in images.

7.2.2 An Adaptive Fusion Architecture for Target Tracking

Face localisation was performed using a vision system that adaptively allocates computational resources over multiple cues to robustly track a target in 3D. A particle filter managed multiple hypotheses of the target location and Bayesian probability theory provided the framework for sensor fusion. Finite computational resources were efficiently allocated across the cues, taking into account the cue's expected utility and resource requirement. The system can accommodate cues running at different frequencies, allowing cues performing less well to be run

slowly in the background for added robustness with minimal additional computation. The system was shown to track a person in 3D space moving in a cluttered environment with variable lighting conditions and occlusions of the target. An additional example was shown demonstrating how the system can be extended to track multiple targets, using multiple particle filters and inhibition of returns to prevent different filters from locking onto the same target. Whilst this system was demonstrated here for robust person tracking, it is equally applicable to tracking other targets.

7.2.3 Facial Feature Detection

A face registration system was developed capable of verifying the presence of a face and automatically detecting facial features in monocular, grey-scale image sequences. The system is able to locate the eyes, mouth corners, nostrils and eyebrows of subjects. The anatomical constraints of the human face are used to govern the feature locating and face verification process. This system both automatically verifies the presence of a face and detects facial feature points for face tracking. The algorithm has been adopted into a commercial face detection system that has seen substantial testing on hundreds of subjects.

7.2.4 3D Deformable Facial Feature Tracking

The problem of tracking deformable facial features was addressed by examining a monocular and a stereo 3D lip tracking system, both operating in conjunction with 3D head trackers to facilitate the head freedom of movement whilst tracking. The monocular system was able to track the head pose and measure the mouth height and width of a subject, whilst the stereo system was capable of tracking the full 3D shape of the mouth as it deformed during speech. Our stereo lip tracking system is the first system that has tracked the raw 3D mouth shape (some other systems have inferred the 3D structure from a monocular view). We do not require a physical model to constrain the shape of the mouth and thus the approach can be easily transferred to tracking other deforming features such as the eyebrows.

7.3 Further Work

We have presented solutions to the problems of localising a face in a scene, detecting facial features, and tracking the face and the motion of facial features. Whilst the systems presented each address particular human-tracking problems, further work is required to integrate these into a single all-inclusive vision system to allow a computer or robot to see a person's face. Such a system would be a valuable tool for enhancing human-machine interaction.

The face localisation system presented in Chapter 4 provides a sound framework for integrating multiple cues with different resource requirements for target tracking. Further work should focus on refining the method of resource allocation and extending the system to adapt online to track an arbitrary and potentially changing number of targets.

Removing the requirement for the subject to blink for the face registration system presented in Chapter 5 would make the algorithm more user friendly as well as allowing it to be employed to detect faces in still images. This improvement has already been implemented in the commercial version of the system implemented by Seeing Machines.

The deformable feature tracking method presented in Chapter 6 is not limited to lip tracking. Extending deformable feature tracking to track eyebrows, nostril shape and other deformable facial features, will allow a computer to really see the deformation of a subject's face. In addition, using colour information has great potential to further increase the accuracy and robustness of the tracking.

However, more interesting than refinements to the systems presented here are the potential applications of this work. Enhancing human-machine interaction is a hot topic of research throughout the world, and computer vision is playing a key role. A computer or robot that can reliably locate and track a person's face in a real world scenario has great potential for application across a wide range of fields. Examples include: improved teleconferencing, monitoring human performance, smart security surveillance, interfaces for the disabled, more realistic interaction for games, and facial animation for digital characters. The possibilities are great, but when we consider how central vision is to the way humans perceive the world it is unsurprising that a computer embodied with this ability offers such potential.

Appendix A

Contents of CD-ROM

The CD-ROM enclosed with this thesis contains the following:

thesis.pdf	Electronic copy of the thesis.
face_images/	Folder containing face images used in Chapter 3 to determine the mean maximum value of O_n .
FRST_movie	Movie of FRST operating over an image sequence.
face_loc_movie1	Movie showing face localisation.
face_loc_movie2	Another movie showing face localisation.
multiple_loc_movie	Movie showing localisation system with multiple targets.
face_reg_movie	Movie showing face registration.
face_track_mono_movie	Movie of monocular lip tracker.
face_track_stereo_movie	Movie of stereo lip tracker.

Appendix B

Derivation of the Optical Flow Constraint Equation

Consider an image sequence $\mathbf{I}(t)$ and denote the intensity at location (x, y) at time t by $\mathbf{I}(x, y, t)$. Using a Taylor series expansion, an expression can be written for $\mathbf{I}(x + dx, y + dy, t + dt)$, that is the value of the image a short time dt later and a small distance (dx, dy) away,

$$\mathbf{I}(x + dx, y + dy, t + dt) = \mathbf{I}(x, y, t) + \frac{d\mathbf{I}}{dx}dx + \frac{d\mathbf{I}}{dy}dy + \frac{d\mathbf{I}}{dt}dt + \dots \quad (\text{B.1})$$

Now, consider an object at a position (x, y) at a time t , that moves through a distance (dx, dy) after time dt . If the intensity of the object in the image does not change, then we have

$$\mathbf{I}(x + dx, y + dy, t + dt) = \mathbf{I}(x, y, t)$$

so it follows from B.1

$$\frac{d\mathbf{I}}{dx}dx + \frac{d\mathbf{I}}{dy}dy + \frac{d\mathbf{I}}{dt}dt = 0$$

dividing through by dt and taking the limit as $dt \rightarrow 0$

$$-\frac{\delta\mathbf{I}}{\delta t} = u\frac{\delta\mathbf{I}}{\delta x} + v\frac{\delta\mathbf{I}}{\delta y} \quad (\text{B.2})$$

where $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$. Equation B.2 is called the optical flow constraint equation.

Bibliography

- Adjoudani, A. and Benoît, C. (1996). *On the integration of audio and visual parameters in an HMM-based ASR*. Springer-Verlag.
- Alexeychuk, A., Di Gesù, V., Palenichka, R., and Valenti, C. (1997). A fast recursive algorithm to compute local axial moments. In *Proc. Converging Computing Methodologies in Astronomy Conference*.
- Aschwanden, P. and Guggenbuhl, W. (1993). Experimental results from a comparative study on correlation-type registration algorithms. In Forstner and Ruweidel, editors, *Robust Computer Vision*, pages 268–289. Wichmann.
- Azarbayejani, A. and Pentland, A. (1995). Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(6), 562–574.
- Azarbayejani, A., Stanner, T., Horowitz, B., and Pentland, A. (1993). Visually controlled graphics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(6), 602–605.
- Bala, L., Talmi, K., and Liu, J. (1997). Automatic detection and tracking of faces and facial features in video sequences. In *Proc. Picture Coding Symposium*.
- Banks, J., Bennamoun, M., and Corke, P. (1997). Fast and robust stereo matching algorithms for mining automation. In *Proceedings of the International Workshop on Image Analysis and Information Fusion*, pages 139–149.
- Baron, J. L., Fleet, D. J., and Beauchemin, S. S. (1994). Performances of optical flow techniques. *Int. Journal of Computer Vision*, **12**(43), 43–77.
- Bates, B. and Cleese, J. (2001). *The Human Face*. British Broadcasting Corporation.

- Benoît, C., Guiard-Marigny, T., Goff, B. L., and Adjoudani, A. (1996). *Which components of the face do humans and machines best speechread?* Springer-Verlag.
- Birdwhistell, R. (1970). *Kinesics and Context*. University of Pennsylvania Press. Philadelphia.
- Cai, J. and Goshtasby, A. (1999). Detecting human faces in colour images. *Image and Vision Computing*, **18**, 63–75.
- Cambridge, A. L. (1994). Database of Faces. [http:// www.cam-orl.co.uk/ face-database.html](http://www.cam-orl.co.uk/face-database.html).
- Chella, A., Di Gesù, V., Infantino, I., Intravaia, D., and Valenti, C. (1999). A cooperating strategy for objects recognition. In *Shape, Contour and Grouping in Computer Vision*, pages 264–276.
- Chen, C. and Chiang, S.-P. (1997). Detection of human faces in colour images. *IEE Proceedings on Vision and Image Processing*, **144**(6), 384–388.
- Chen, T. (2001). Audiovisual speech processing. *IEEE Signal Processing Magazine*, pages 9–21.
- Chuang, M.-M., Chang, R.-F., and Huang, Y.-L. (2000). Automatic facial feature extraction in model-based coding. *Journal of Information Science and Engineering*, **16**, 447–458.
- Clark, A. F. and Kokuer, M. (1992). Feature identification and model tracking. In *11th IAPR International Conference on Pattern Recognition*, volume 3, pages 79–82.
- Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt¹, P., and Wixson¹, L. (2000). A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, The Robotics Institute, Carnegie Mellon University.
- Crowley, J. and Coutaz, J. (1995). Vision for man machine interaction.
- Crowley, J. L. and Berard, F. (1997). Multi-modal tracking of faces for video communications. In *Computer Vision and Pattern Recognition, CVPR '97*.

- Darrell, T., Gordon, G., Harville, M., and Woodfill, J. (2000). Integrated person tracking using stereo, color, and pattern recognition. *International Journal of Computer Vision*, **37**(2), 175–185.
- Di Gesù, V. and Palenichka, R. (2001). A fast recursive algorithm to compute local axial moments. *Signal Processing*, **81**, 265–273.
- Di Gesù, V. and Valenti, C. (1995a). The discrete symmetry transform in computer vision. Technical Report DMA 011 95, Palermo University.
- Di Gesù, V. and Valenti, C. (1995b). Symmetry operators in computer vision. In *Proc. First CCMA Workshop on 'Vision Modeling and Information Coding'*.
- Di Gesù, V., C., V., and L, S. (1996). Local operators to detect regions of interest. *Pattern Recognition Letters*.
- Dodd, B. and Campbell, R. (1987). *Hearing by Eye: The Psychology of Lipreading*. Lawrence Earlbaum Associates Ltd.
- Duda, R. and Hart, P. (1972). Use of the hough transform to detect lines and curves in pictures. *Communications of the Association for Computing Machinery*, **15**(1), 11–15.
- Fletcher, L., Apostoloff, N., Chen, J., and Zelinsky, A. (2001). Computer vision for vehicle monitoring and control. In *Proc. of Australian Conference on Robotics and Automation*, Sydney, Australia.
- Gee, A. and Cipolla, R. (1994). Non-intrusive gaze tracking for human-computer interaction. In *Proc. of the International Conference on Mechatronics and Machine Vision in Practice*, pages 112–117.
- Goecke, R. (2002). *PhD thesis*. Ph.D. thesis, Australian National University.
- Goecke, R., Miller, J. B., Zelinsky, A., and Robert-Ribes, J. (2000). Automatic extraction of lip feature points. In *Australian Conference on Robotics and Automation*, pages 31–36.
- Graf, H. P., Chen, T., Perajan, E., and Cosatto, E. (1995). Locating faces and facial parts. In *International Workshop on Automatic Face and Gesture Recognition*, pages 41–46.

- Grammer, K. and Thornhill, R. (1994). Human facial attractiveness and sexual selection: The roles of averageness and symmetry. *Journal of Comparative Psychology*, **108**(3), 233–242.
- Hakkanen, H., Summala, H., Partinen, M., Tiihonen, M., and Silvo, J. (1999). Blink duration as an indicator of driver sleepiness in professional bus drivers. *Sleep*, **22**(6), 798–802.
- Hartley, R. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge University Press.
- Hartley, R. I. and Sturm, P. (1995). Triangulation. In *Proceedings of Conference on Computer Analysis of Images and Patterns*.
- Hjelmås, E. and Low, B. K. (2001). Face detection: A survey. *Computer Vision and Image Understanding*, **83**, 236–274.
- Horn, B. K. P. (1986). *Robot Vision*. McGraw Hill.
- Horn, B. K. P. and Schunk, B. G. (1981). Determine optical flow. *Artificial Intelligence*, **17**, 185–203.
- Hunke, H. M. (1994). Locating and tracking of human faces with neural networks. Technical Report CMU-CS-94-155, Carnegie Mellon University.
- Hunke, M. and Waibel, A. (1994). Face locating and tracking for human-computer interaction.
- Intrator, N., Reisfeld, D., and Yeshurun, Y. (1995). Extraction of facial features for recognition using neural networks.
- Isard, M. and Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In *Proc. of European Conf. on Computer Vision*, volume 1, pages 343–356.
- Isard, M. and Blake, A. (1998). Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, **29**(1), 5–28.
- Jarvis, R. (1983). A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 122–139.

- Kagami, S., Okada, K., Inaba, M., and Inoue, H. (1999). Real-time 3d flow generation system. In *IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*.
- Kagami, S., Okada, K., Inaba, M., and Inoue, H. (2000). Design and implementation of onbody real-time depthmap generation system. In *IEEE Int. Conf. on Robotics and Automation*, California, USA. IEEE Computer Press.
- Kanade, T. (1973). Picture processing by computer complex and recognition of human faces. Technical report, Kyoto University, Dept. of Information Science.
- Kanade, T., Yoshida, A., Oda, K., Kano, H., and Tanaka, M. (1996). A stereo machine for video rate dense depth mapping and its new applications. In *Proc. of International Conference on Computer Vision and Pattern Recognition.*, pages 196–202.
- Katahara, S. and Aoki, M. (1999). Face parts extraction window based on bilateral symmetry of gradient direction. In *Proceedings of 8th International Conference on Computer Analysis of Image Patterns, CAIP'99*, pages 489–497.
- Kaucic, R., Dalton, B., and Blake, A. (1996). Real-time lip tracking for audiovisual speech recognition applications. In *Proc European Conference on Computer Vision*, pages 376–387.
- Kaufman, L. and Richards, W. (1969). Spontaneous fixation tendencies of visual forms. *Perception and Psychophysics*, **5**(2), 85–88.
- Kim, S.-H. and Kim, H.-G. (2000). Face detection using multimodal information. In *Proc. of IEEE International Conference on Face and Gesture Recognition*, pages 14–19.
- Kimme, C., Ballard, D., and Sklansky, J. (1975). Finding circles by an array of accumulators. *Communications of the Association for Computing Machinery*, **18**(2), 120–122.
- Konolige, K. (1997). Small vision system: Hardware and implementation.
- Kotropoulos, C. and Pitas, I. (1997). Rule-based face detection in frontal views. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 21–24.

- Kovesi, P. (1997). Symmetry and asymmetry from local phase. In *Tenth Australian Joint Conference on Artificial Intelligence*.
- Kovesi, P. (1999a). Image features from phase congruency. *Videre: Journal of Computer Vision Research*, **1**(3).
- Kovesi, P. (1999b). Matlab code for calculating phase congruency and phase symmetry/asymmetry. <http://www.cs.uwa.edu.au/~pk/Research/phasecong.m>.
- Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, **22**(1), 79–86.
- Kumar, V. P. and Poggio, T. (2000). Learning-based approach to real time tracking and analysis of faces. In *Proc. of IEEE International Conference on Face and Gesture Recognition*, pages 96–101.
- Lin, C.-C. and Lin, W.-C. (1996). Extracting facial features by an inhibitory mechanism based on gradient distributions. *Pattern Recognition*, **29**(12), 2079–2101.
- Locher, P. and Nodine, C. (1987). Symmetry catches the eye. In J. O'Regan and A. Lévy-Schoen, editors, *Eye Movements: from physiology to cognition*. Elsevier Science Publishers B. V.
- Lowe, D. G. (1991). Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**(5), 441–450.
- Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proc. DARPA image understanding workshop*, pages 121–130.
- MacCormick, J. and Blake, A. (1998). A probabilistic contour discriminant for object localisation. In *Proc International Conference on Computer Vision*.
- Matsumoto, Y. and Zelinsky, A. (2000). An algorithm for real-time stereo vision implementation of head pose and gaze direction estimation. In *Proc. of Workshop on Automatic Face and Gesture Recognition*, pages 499–504.
- Maurer, T. and von der Malsburg, C. (1996). Tracking and learning graphs on image sequences of faces. In C. v.d. Malsburg, W. v. Seelen, J. Vorbrüggen, and B. Sendhoff, editors, *Proceedings of the ICANN 1996*, pages 323–328, Bochum.

- McGurk, H. and MacDonald, J. (1976). Hearing lips and seeing voices. *Nature*, **264**, 746–748.
- Mehrabian, A. (1972). *Non-verbal communication*. Aldine-Atherton, Chicago, Illinois.
- Minor, L. G. and Sklansky, J. (1981). Detection and segmentation of blobs in infrared images. *IEEE Transactions on Systems Man and Cybernetics*, **SMC-11**(3), 194–201.
- Nagel, H. H. (1983). Displacement vectors derived from second-order intensity variation in image sequences. *Computer Graphics and Image Processing*, **21**, 85–117.
- Newman, R., Matsumoto, Y., Rougeaux, S., and Zelinsky, A. (2000). Real-time stereo tracking for head pose and gaze estimation. In *Proc. of the Fourth International Conference on Face and Gesture Recognition*, pages 122–128.
- Oliver, N., Pentland, A., and Bèrard, F. (1997). Lafter: Lips and face real time tracker. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR97)*.
- Omara, D. (2000). Skin colour modelling. In *Proceedings of the Tenth University of Western Australia Computer Science Research Conference*, pages 113–139.
- Palenichka, R. M., Zaremba, M. B., and Valenti, C. (2001). A fast recursive algorithm for the computation of axial moments. In *Proc of the 11th International Conference on Image Analysis and Processing*, pages 95–100.
- Petajan, E. D. (1985). Automatic lipreading to enhance speech recognition. In *IEEE Computer society conference on computer vision and pattern recognition*, pages 40–47.
- Pheasant, S. (1986). *Bodyspace Anthropometry, Ergonomics and Design*. Taylor and Francis Ltd.
- Proffitt, D. and Cutting, J. (1980). Perceiving the centroid of curvilinearly bounded rolling shapes. *Perception and Psychophysics*, **28**(5), 484–487.
- Raha, Y., McKenna, S., and Gong, S. (1998). Tracking and segmenting people in varying lighting conditions using colour. In *Proceedings of International Conference on Automatic Face and Gesture Recognition*.

- Reinders, M. J. T., Sankur, B., and van der Lubbe, J. C. A. (1992). Transformation of a 3d facial model into an actual scene face. In *11th IAPR International Conference on Pattern Recognition*, volume 3, pages 75–78.
- Reisfeld, D. (1993). *Generalized Symmetry Transforms: Attentional Mechanisms and Face Recognition*. Ph.D. thesis, Tel-Aviv University.
- Reisfeld, D. and Yeshurun, Y. (1998). Preprocessing of face images: Detection of features and pose normalisation. *Computer Vision and Image Understanding*, **71**(3), 413–430.
- Reisfeld, D., Wolfson, H., and Yeshurun, Y. (1995). Context free attentional operators: the generalized symmetry transform. *International Journal of Computer Vision, Special Issue on Qualitative Vision*, **14**, 119–130.
- Research, B. T. (2001). The BioID Face Database. [http:// www.bioid.de/ bioid-db.zip](http://www.bioid.de/bioid-db.zip).
- Revéret, L. and Benoît, C. (1998). A new 3d lip model for analysis and synthesis of lip motion in speech recognition. In *Proceedings of the International Conference on Auditory-Visual Speech Processing, AVSP98*, pages 207–212.
- Richards, W. and Kaufman, L. (1969). Centre-of-gravity tendencies for fixations and flow patterns. *Perception and Psychophysics*, **5**(2), 81–84.
- Satoh, S., Nakamura, Y., and Kanade, T. (1997). Name-it: Naming and detecting faces in video by the integration of image and natural language processing.
- Schiele, B. and Waibel, A. (1995). Gaze tracking based on face color. In *International Workshop on Automatic Face and Gesture Recognition*.
- Sela, G. and Levine, M. D. (1997). Real-time attention for robotic vision. *Real-Time Imaging*, **3**, 173–194.
- Shakunaga, T., Ogawa, K., and Oki, S. (1998). Integration of eigentemplate and structure matching for automatic facial feature detection. In *Proc. International Conference on Automatic Face and Gesture Recognition (FG'98)*, pages 94–99.
- Sigal, L. and Sclaroff, S. (2000). Estimation and prediction of evolving color distributions for skin segmentation under varying illumination. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2000)*.

- Sobottka, J. and Pitas, I. (1996a). Segmentation and tracking of faces in color images. In *Proceedings of the Second International Conference on Automation Face and Gesture Recognition*, pages 236–241.
- Sobottka, K. and Pitas, I. (1996b). Looking for faces and facial features in color images.
- Soto, A. and Khosla, P. (2001). Probabilistic adaptive agent based system for dynamic state estimation using multiple visual cues. In *To appear in Proceedings of International Symposium of Robotics Research (ISRR)*.
- Stern, J. (2002). Why do we blink? <http://www.msnbc.com/news/244710.asp>.
- Sun, Q. B., Huang, W. M., and Wu, J. K. (1998). Face detection based on color and local symmetry information. In *Proceedings of Third International Conference on Face and Gesture Recognition*, pages 130–135.
- Swain, M. J. and Ballard, D. (1991). Color indexing. *International Journal of Computer Vision*, **7**(1).
- Terrillon, J.-C. and Akamatsu, S. (1999). Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images. In *Vision Interface '99 (VI'99)*.
- Terrillon, J.-C., David, M., and Akamatsu, S. (1998). Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments. In *Proc. of the Third International Conference on Automatic Face and Gesture Recognition*, pages 112–117.
- Thrun, S. (2000). Probabilistic algorithms in robotics. *AI Magazine*, **21**(4), 93–109.
- Triesch, J. and von der Malsburg, C. (2000). Self-organized integration of adaptive visual cues for face tracking. In *Proc. of IEEE International Conference on Face and Gesture Recognition*, pages 102–107.
- Trucco, E. and Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Prentice Hall.
- Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, **3**(1), 71–86.

- Varchmin, A. C., Rae, R., and Ritter, H. (1997). Facial feature detection using neural networks. In *Proceedings of International Conference on Artificial Neural Networks*.
- Wu, H., Chen, Q., and Yachida, M. (1999). Face detection from color images using a fuzzy pattern matching method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**(6), 557–563.
- Xu, M. and Akatsuka, T. (1998). Detecting head pose from stereo image sequence for active face recognition. In *Proc. of 3rd International Conference on Face and Gesture Recognition (FG98)*.
- Yamamoto, H., Yeshurum, Y., and Levine, M. (1994). An active foveated vision system: attentional mechanisms and scan path convergence measures. *CVGIP: Image Understanding*.
- Yang, J. and Waibel, A. (1996). A real-time face tracker. In *Proceedings of the Third Workshop on Applications of Computer Vision*, pages 142–147.
- Yang, J., Stiefelhagen, R., Meier, U., and Waibel, A. (1998a). Real-time face and facial feature tracking and applications. In *Proceedings of Auditory-Visual Speech Processing (AVSP 98)*.
- Yang, J., Lu, W., and Waibel, A. (1998b). Skin-color modeling and adaptation. In *Proceedings of ACCV'98*, volume II, pages 687–694.
- Yang, M.-H. and Ahuja, N. (1998). Detecting human faces in color images. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 127–130.
- Yang, M.-H., Ahuja, N., and Kriegman, D. (2000). Face detection using a mixture of linear subspaces. In *Proc. of IEEE International Conference on Face and Gesture Recognition*, pages 70–88.
- Yang, M.-H., Kriegman, D., and Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(1), 34–58.
- Yow, K. C. and Cipolla, R. (1995). Towards an automatic human face localization system. In *Proceedings of 5th British Machine Vision Conference*, volume 2, pages 701–710.

- Yow, K. C. and Cipolla, R. (1997). Feature-based human face detection. *Image and Vision Computing*, **15**(9), 713–735.
- Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In *3rd European Conf. Computer Vision, Stockholm*.
- Zarit, B., Super, B., and Quek, F. (1999). Comparison of five color models in skin pixel classification. In *Proc. International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 58–63.