Learning and Initialisation for Visual Tracking



A. P. Wildenberg
Robotics Research Group
Department of Engineering Science
University of Oxford

Supervisor: Prof. A. Blake

June 7, 1998

This thesis is submitted to the Department of Engineering Science, University of Oxford, as partial completion of the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise indicated, describes my own research.

Abstract

The research described in this thesis describes work towards making Active Contours fully autonomous mechanisms capable of tracking biological targets in real-world situations. The problem of tracking the apparent motion of cauliflower plants photographed from a tractor in a field was of special interest. In order to reach this simple goal it was necessary to complete research in a number of different areas.

Active Contours use a stochastic model of motion to accurately predict the likely location of a target given previous observations. The standard method to acquire the parameters of this model is to learn them from training sequences. These learnt models track more accurately and robustly than hand-tuned models. Implicit in previous learning algorithms is the assumption that the same object is tracked in all the training sequences, and that after training the same object will be tracked when the new model is used. Many times it is useful to track a class of objects instead of a particular object. When tracking classes of objects, there are two sources of variation of the shape as observed in an image: change due to the motion of a particular target, and change due to different targets having different shapes. When these two sources of variation are modelled as one, the model cannot adequately model both, and the robustness of the tracking is reduced.

To allow the robust tracking of classes of objects, a new, augmented, model for tracking is introduced. A novel learning algorithm is presented that can learn the parameters of the augmented model through maximum likelihood estimation. The learning algorithm also correctly learns with short training sequences, a problem with the previous learning algorithm. The new model with learnt dynamics is shown to track more robustly than the previous model with learnt dynamics.

An area of research important to making Active Contour trackers autonomous is the initialisation problem. In the past, the initialisation problem has been solved by the operator manually positioning the curve, and then starting the tracker with a large initial uncertainty. To initialise the contour automatically, three steps are necessary:

Firstly, it is necessary to locate features of the target in the image. One way to do this is to segment the image into the target and background. A variety of pixel-based and region-based classification schemes are investigated for this. While the data provided by the segmentation in general is not sufficient for good tracking, it is possible to use this segmentation to recover the initial location of the target.

Once the target has been identified, it is necessary to recover the pose of the target. It is shown that there is no general solution to this problem. However, when the motion models have certain forms, it is possible to completely recover the pose. Methods are presented for recovering affine transformations and transformations contained within the affine space. For more general models, such as those based on PCA, methods are presented for partial pose recovery. Both are evaluated on both real and synthetic data.

The last step of Active Contour initialisation is determining the accuracy/uncertainty of the pose recovery. Previously known and novel methods to do this are analysed from a probabilistic standpoint. One novel method is presented which allows the initial uncertainty to be learned from training sequences in a similar manner to the way the motion model is

learned. The problem of simultaneously estimating the initial pose and uncertainty is also discussed briefly.

Throughout the thesis, both theoretical arguments and experimental results are presented to support the work completed.

"The time has come," the Walrus said,
"To talk of many things:
Of shoes – and ships – and sealing-wax –
Of cabbages – and kings –
And why the sea is boiling hot –
And whether pigs have wings."

Lewis Carroll, *Through the Looking Glass*

Acknowledgments

To my supervisor, Dr. Andrew Blake, for his help and guidance throughout the course of the work in this thesis.

To Dr. David Reynard for introducing me to the rites of Kalman filtering and helping me see cabbages as moving objects.

To Dr. Simon Rowe for convincing me it was possible to finish and providing me with lots of excuses to go to George and Davis.

To the Oxford Tracking Group (Dr. Benedicte Bascle, Colin Davidson, Major Robert Kaucic, Michael Isard, Ben North) for countless hours of discussion, fights and proof-reading.

To my family for being my family.

To Charles R. B. Day for accepting that I don't row.

And to Dr. Carrie La Seur for absolutely, positively everything.

Contents

Acknowledgments ii

1 Introduction 1
- 1.1 Robots and the environment . . . 2
- 1.2 Automatic tracking systems . . . 4
- 1.3 Overview of this document . . . 5

2 Literature Survey 8
- 2.1 Image Processing and Pattern Recognition . . . 8
- 2.2 Kalman Filtering . . . 9
- 2.3 Active Vision . . . 10
  - 2.3.1 Foveal vision systems and active contours . . . 11
- 2.4 Visual tracking . . . 13
  - 2.4.1 Flow based tracking . . . 13
  - 2.4.2 Feature based tracking . . . 15
    - Corner tracking . . . 15
    - Other feature tracking . . . 16
- 2.5 Agrotics and computer vision . . . 16
  - 2.5.1 Agrotics . . . 17
  - 2.5.2 Computer vision in agriculture . . . 17
  - 2.5.3 Combined computer vision/agrotic systems . . . 18
- 2.6 Criticisms of previous work . . . 21

3 Colour Based Pattern Recognition 22
- 3.1 Measuring minimal risk for visual data . . . 22
  - 3.1.1 Fundamentals of pattern classification . . . 23
  - 3.1.2 Classifier training . . . 24
  - 3.1.3 Evaluating measurement techniques by finding an upper limit of accuracy . . . 26
  - 3.1.4 Summary . . . 27
- 3.2 Different colour space representations for visual data representation . . . 27
  - 3.2.1 Visible light, black and white "colour" space . . . 29
  - 3.2.2 RGB colour space . . . 30

3.2.3 HSV colour space 31
3.2.4 Other representations of colours in digital images 32
3.2.5 Summary 32
3.3 The Dichromatic Reflection Model 34
3.3.1 A mathematical description of the DRM 34
3.3.2 The dichromatic reflection model for image understanding 36
3.3.3 Implications of the dichromatic reflection model on the choice of colour representation 36
3.3.4 Conclusion 37
3.4 Summary 39

4 Applications of colour based pattern recognition 40
4.1 Data preparation 40
4.2 Classification of field cauliflowers in monochrome images 41
4.3 Classification of field cauliflowers using RGB space 44
4.3.1 Upper limit of accuracy 44
4.3.2 A simple linear classifier in RGB space 45
Statistical analysis of data 46
4.3.3 A linear classifier in RGB space 48
4.3.4 Summary 50
4.4 Classification of field cauliflowers in hue space 50
4.4.1 An efficient algorithm for finding minimal risk in hue space 52
4.4.2 Application of posterior white balancing to decrease risk of hue-based classification 53
4.5 Improving classification through local information 55
4.6 Analysis of Results 58
4.6.1 Description of the classifiers tested 58
4.6.2 Comparison of performance 59
4.7 Conclusion 61

5 Modelling shape and motion independently in an Active Contour tracker 62
5.1 Introduction 62
5.2 Active Contours 63
5.3 Learning general shape spaces 69
5.3.1 Building general shape spaces through keyframes 69
5.3.2 Refining general shape spaces through PCA 72
5.3.3 Results of learning shape spaces on real data 74
5.4 Learning dynamics over short data sequences 77
5.4.1 The mean of the sequence is not the mean of the system 78
5.4.2 Learning algorithm for single short sequences 79
5.5 Learning dynamics from multiple training sequences 81

5.6 Shape and motion variability 83
5.7 Modelling class and motion variability 85
5.7.1 Extended dynamical model 89
5.7.2 Learning algorithm 89
5.7.3 Recovery of $\overline{\mathbf{X}}^{\gamma}$ 94
5.7.4 Observability and controllability of the augmented state equations . 94
5.7.5 System reformulation to remove singularities in $I - A_0 - A_1$ 97
5.8 Results 102
5.9 Future Work 103
5.9.1 Pruning the dimension of the Kalman filter 103

6 The Initialisation Problem and Active Contours 108
6.1 B-splines and parameter dependences 110
6.1.1 A parameterisation invariant distance function 110
6.1.2 Fitting B-Splines to arbitrary curves 114
6.2 Initialisation of the Kalman filter 115
6.2.1 No initialisation 116
6.2.2 Motion alarms 116
6.2.3 Automatic recovery of the starting state 117
6.2.4 Moment based initialisation 117
6.2.5 The canonical frame 122
6.2.6 Statistically approximating the sixth d.o.f. through prior information 124
6.2.7 Second order moments and a distinguished point 125
6.2.8 Backprojection of fit for sixth d.o.f. 126
6.2.9 Using higher order moments 127
6.2.10 Reduced dimensionality solutions 139
6.2.11 Optimal subspaces within affine spaces 140
6.2.12 Invariant initialisation 142
6.3 Moment-based initialisation with general shape spaces 145
6.3.1 Using the affine initialisation for general shape spaces 146
6.4 Numerical solutions for initialising general shape spaces 147
6.4.1 Five d.o.f. initialisation in a general state space 148
6.4.2 Separating translation from other degrees of freedom 148
6.5 Summary 151

7 Initialisation of the state uncertainty 153
7.1 Estimating uncertainty from learnt dynamics 154
7.1.1 Estimating initial uncertainty from training sequences 156
7.1.2 Estimating initial uncertainty for an initialiser 159
7.2 Combining initialisation of state estimate and initialisation of uncertainty . 159
7.3 Summary 161

8 Conclusion and future work **163**
8.1 Automatic redimensioning of the Kalman filter 166
8.2 Initialisation of the Active Contour in the absence of segmentation 168

A Additional mathematical derivations and formulae **170**
A.1 Kalman filter state equations 170
A.2 Formulae for moments of B-splines 171
A.3 Some lemmas about matrix and vector differentiation 173
A.4 Using homogeneous equations to simplify initialisation 174

1

Introduction

When computer vision began to emerge as a field of its own thirty years ago, there were high expectations all around. But it wasn't very long before researchers began to realise that it was a lot more difficult to see than people had noticed before. Every day humans effortlessly perform image segmentation, object recognition, pose recovery, auto-calibration, colour and texture segmentation and recognition, as well as many other tasks that have been assigned to the research area of computer vision. After thirty years of research, there are still few visually-based tasks that computers can execute as well as or better than humans.

For a long time, most progress in computer vision research was made by simplifying the world which the computers had to analyse. The first recognition and pose-recovery experiments were performed on 'block-worlds' – closed environments with polyhedral objects and uniform lighting conditions. As methods and their corresponding algorithms progressed, the conditions under which the methods worked were gradually made more and more like those in the real world. Computer vision has finally begun to mature to the point where some of its contributions are useful for solving real-world problems.

1.1 Robots and the environment

One field in which computer vision is starting to have an impact is agricultural robotics, or agrotics (Marchant, 1991). The environmental impact of herbicides and other chemicals used in agriculture has become a serious concern in recent years. Pesticides and herbicides have become an indispensable weapon in the farmer's constant drive to increase production quality and efficiency. Approximately 23,000 tonnes of chemicals are used annually in the UK at a cost of more than £400 million (Blake and Marchant, 1992) plus many additional hidden costs. The chemicals contaminate the water supply and disrupt the food chain. Residual farm chemicals remain in plants, meat and milk brought to market.

The precision application of chemicals in fields could help reduce the tonnage of chemicals applied while maintaining or improving the current level of crop protection. Oxford University and the Silsoe Research Institute have begun a collaborative clean technology programme, of which this research is a part, whose goal is to produce a small autonomous guided vehicle capable of passing between rows of crops and performing precision application of chemicals to plants it has previously recognised (Blake and Marchant, 1992). Such a robot would combine state of the art computer vision, autonomous vehicle guidance and robotics in a practical and useful application (see figures 1.1 and 1.2).

A large part of the research in this thesis was motivated by the requirements of this vehicle's computer vision system. To apply chemicals precisely to plants in the field, the plants must first be recognised as being distinct from the soil. Then they must be categorised into weeds or crops. The problem of classifying the plants is far from trivial. Even when restricted to the simple case of a field of drilled cauliflower, there are many variables. Drilled cauliflower are planted in the field on a grid in evenly spaced rows and columns. The general location of the cauliflower is known, but the grid is only approximate and contains holes where cauliflower have been missed by the planter or where plants have died. To further complicate recognition, each cauliflower has a unique size and shape which changes during

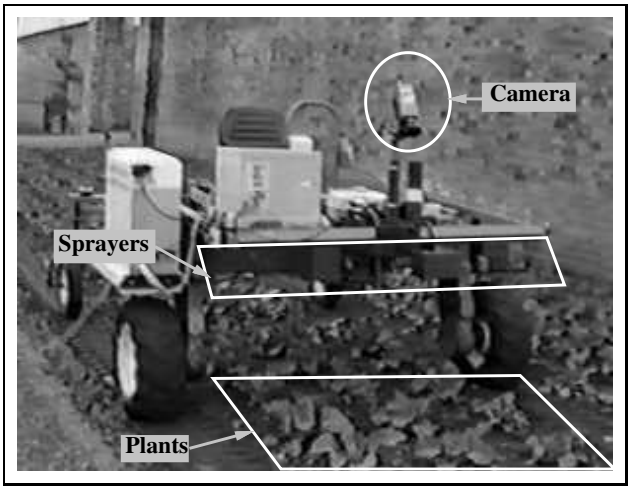


Figure 1.1: **A tractor for precision agriculture.** *An autonomous tractor developed by the Silsoe Research Institute incorporates a downward pointing camera to monitor plants passing under an array of spray nozzles. Sprayed chemicals can then be directed onto plants, or to avoid plants, as appropriate. Photograph courtesy of Silsoe Research Institute.*



Figure 1.2: **Plant tracking.** *A mobile vehicle sees plants (cauliflower in this instance) passing through the field of view of a downward pointing camera. The plant's motion and shape is captured by a dynamic contour, shown in white.*

the lifetime of the plant.

The problem of recognising weeds is even harder, since there are so many different shapes and sizes of weeds that can appear in a given field. If model-based vision techniques are to be used to aid in recognition, the models employed must be able to describe the natural diversity of the plants.

The changing weather brings with it diverse lighting conditions which make image recognition and understanding more difficult than in controlled laboratory conditions. For practical reasons, artificial lighting should be avoided if at all possible. To make tracking more robust, a model of natural illumination is needed to predict the effects that the change in natural lighting will have on the scene.

Images taken in the cauliflower fields show very poor contrast in visible light monochrome (i.e. standard black and white) images. Other sources of video information, such as colour cameras and near infra-red cameras provide better contrast. In the case of segmented and or colour images, preliminary image processing and pattern recognition is necessary before more traditional vision techniques can be employed.

Once plants have been identified in an image, the physical location of the plant must be recovered in order to coordinate the spraying mechanism of the robot. The plant must also be tracked in order to provide velocity information for the guidance systems of the vehicle.

1.2 Automatic tracking systems

Visual tracking is a field of computer vision that is starting to make great strides. For many years it was difficult to do real-time tracking because of the sheer volume of data that needed to be processed. Recent algorithms such as Snakes (Kass et al., 1987), Active Shape Models (Cootes and Taylor, 1992), and Active Contours (Blake et al., 1993b) have addressed this problem by carefully choosing representations that identify regions of interest and features within those regions of interest with very low computational overhead. These algorithms, coupled with significant advances in the capabilities of hardware have made

real-time tracking on standard computer workstations a reality.

Model-based trackers, such as Active Contours, rely on strong *a priori* models of shape and motion. Many times in the past, these models have been tailored to optimise tracking performance for a particular object, for example William Jefferson Clinton's right hand. A much more useful model of shape and motion would be optimised for a class of objects, for example human hands. In order to increase the generality of the models to handle classes of objects, it is important that they be explicitly modelled.

One of the last barriers to making visual tracking a commercially viable technology is that most trackers, in some form or another, rely on human assistance. The essence of the tracking problem is taking past knowledge about the location and dynamical motion of a target and combining that knowledge with information that has been gathered about the present state of the target. The initialisation problem, that of locating the target to track for the first time, is much more difficult than the normal (pre-initialised) tracking problem. As a result, it has been neglected as an area of research. In the absence of automatic initialisation techniques, humans are required to start trackers, thus greatly decreasing the usefulness of the system.

1.3 Overview of this document

The aim of the work in this thesis is to extend and adapt the curve trackers described by Blake (Blake et al., 1993b) to handle the problem of tracking biological subjects in real-world situations, such as those that would occur during the operation of the Silsoe autonomous tractor. To achieve this goal requires advances in three general areas: the basic pattern recognition to recognise the plants in the images, the extension of the Active Contour mechanism to model classes of objects separately from individuals of that class, and techniques for the automatic initialisation of the Active Contour tracker. The structure of the thesis follows the same order of topics.

Literature review Chapter two reviews the general background literature for tracking and pattern recognition. Its goal is to cast the new research in light of what has previously been accomplished in these fields.

Colour-based pattern recognition Chapter three develops some basic theory about pattern recognition in colour images. The mathematics of Bayesian classification and pattern matching are developed, as well as a theoretical limit for all classifiers given a particular training set. The different representations of colour images are discussed especially with respect to the Dichromatic Reflection Model, a model that relates the perceived colour in an image to the colour of the surface of the object and the light of the illuminating source.

Chapter four takes this theoretical basis and applies it to the problem of segmenting images of plants taken from the Silsoe tractor under suboptimal but realistic conditions.

Modelling shape and motion independently in an Active Contour tracker Once the framework for recognising the plants has been laid, chapter five provides extensions to the Active Contour framework to handle tracking classes of objects. The apparent motion of a plant as observed from a moving vehicle is very simple. The differences between a plant and its neighbour can be quite complex. The changes in appearance resulting from these two causes were modelled as one in previous Active Contour systems. Chapter five develops an extended framework that explicitly separates the two sources of variation, and provides analysis of the observability and controllability of the resulting model.

The initialisation problem and Active Contours For the tracking system to be fully autonomous, it is critical that the system be able to start and stop itself. While stopping a tracker and detecting that a new plant has entered the field of view are relatively simple problems, determining the initial pose of a plant that has just entered the field of view is hard. Techniques are discussed in chapters six and seven for approximating this initial pose as well as determining the uncertainty of that estimate, both of which are needed for the

initialisation of the Kalman filter used by the Active Contours.

Applications Applications of the research are presented through the thesis. The pattern recognition theory is tested on real images taken by the Silsoe tractor. The separation of shape and motion variability is shown to be robust on real data that has been modified to be especially difficult to track. The techniques for determining the initial pose of plants are tested on real images.

Contributions In addition to providing information about solving the particular problems encountered with the Silsoe vehicle, this thesis makes several important contributions to the general field of computer vision and, in particular, visual tracking. The extended model developed in chapter five enables the Active Contour framework to be used to track biological systems that previously had been considered too complex. The discussion of initialisation in chapters six and seven describes new methods for determining the initial pose of objects, and provides one of the few discussions of initialising Kalman filters available in the literature.

Constructing the visual processing for an autonomous robot designed to operate in agriculture may be a very specific task, but the new research required to reach that goal is substantial.

2

Literature Survey

This chapter reviews some of the previous work done in fields related to the research in this thesis.

2.1 Image Processing and Pattern Recognition

While similar to computer vision, image processing and pattern recognition differ from computer vision in both their goals and approaches. Image processing serves two basic purposes: improving pictorial information for human interpretation, and processing scene data for autonomous machine perception (Gonzales and Woods, 1992). In the classic textbook *Pattern Classification and Scene Analysis*, Duda and Hart (Duda and Hart, 1973) define pattern classification as the assignment of a physical object or event to one of several prespecified categories. According to Nalwa (Nalwa, 1993), the emphasis in pattern recognition is on two-dimensional patterns, such as the letters of the alphabet, whereas computer vision is concerned with generating descriptions of three-dimensional scenes not constrained to be members of predetermined sets from two-dimensional images of the scenes.

In Duda and Hart, the fundamental theory of pattern recognition is covered in detail.

A description of a general full pattern classification system is given. A statistical model of classification is developed for the problem in decision theoretic terms, and the Bayes decision rule, an optimal classifier when all the probability distributions of the problem are known, is developed. Later, parametric classifiers are introduced for problems when the p.d.f.'sare unknown. Classifiers covered include n-nearest neighbour algorithms, Fisher's optimal linear discriminant, perceptron (neural network) classifiers, and stochastic approximation (simulated annealing). Additionally, algorithms for unsupervised learning and clustering are described.

2.2 Kalman Filtering

The Kalman filter (Bucy, 1969) is an optimal linear filter when the following two conditions hold: (1) the process model is represented by a linear differential equation corrupted by additive Gaussian noise. (2) the measurements are a linear function of the (unknown) states corrupted by additive Gaussian noise. Under these conditions, the prior of the state is Gaussian and can be represented by a mean vector (the state estimate) and a covariance matrix (the uncertainty). By representing this information in different ways it is possible to decrease the sensitivity of the Kalman filter to numerical error (the square root filter), or let the filter represent infinite uncertainty (the information filter). If the measurement model and process model are invariant over time, it is possible to significantly reduce the computational complexity of the filter by setting the Kalman gain (the influence each measurement has on the prior estimate) to a constant (the steady-state Kalman filter or $\alpha - \beta$ filter (Bar-Shalom and Fortmann, 1988)).

If the underlying dynamics of the system are non-linear, various derivations of the Kalman filter can be used for estimation. The Extended Kalman Filter (Bar-Shalom and Fortmann, 1988; Grewal and Andrews, 1993) linearises the equations for state update and measurement integration at each timestep through a first-order Taylor series expansion. The Iterated Extended Kalman Filter performs multiple first-order Taylor series expan-

sions until the system update and measurement equations converge. For systems of very high dimensionality, it is possible to approximate the Kalman filter using data reduction techniques, resulting in the singular value decomposition filter, and the partial eigendecomposition Kalman filter (Cohn and Todling, 1995).

All of these derivatives of the Kalman filter represent the prior probability of the state with a mean vector and a covariance matrix. In the case of non-linear state equations or measurement models, the prior of the state is almost never Gaussian. This causes an error in the estimation of the prior (Maybank et al., 1996). There are several filters that have been designed as non-linear filters from the ground up. The Gaussian sum filter (Bar-Shalom and Fortmann, 1988; Reynard, 1993) represents the prior of the state with a weighted sum of Gaussians. The Condensation filter (Isard and Blake, 1996) models the prior of the state variable by a set of samples. At each timestep, samples are randomly chosen, allowed to diffuse forward according to the state noise model, then checked for support by the measurements.

The Kalman filter and its various derivatives have become standard tools in computer vision (Dickmanns and Graefe, 1988; Stephens, 1990; Lowe, 1992; Gennery, 1992; Harris, 1992b; Terzopoulos and Szeliski, 1992; Blake et al., 1993b), as well as such fields as vehicle control (Durrant-Whyte, 1988), and weather prediction (Cohn and Todling, 1995).

2.3 Active Vision

Most computer vision systems are of an inherently passive nature. They make measurements on the world visible through the camera, but do not interact with it. Many computer vision systems reduce the streams of visual information humans perceive to isolated images. When humans view a scene they perceive to be difficult to understand, they can change their viewpoint in a way that will provide disambiguating information, but because most machine vision systems have no control over their extrinsic (or even intrinsic) parameters, they must extract their measurements from a system whose geometry may not be ideal.

According to Blake and Yuille (1992) Active Vision refers not to sensing technology but to strategies for observation. While the definition may vary, Active Vision implies that the computer vision process uses previously acquired information to influence the present state of the world around it with the goal of improving future observations.

2.3.1 Foveal vision systems and active contours

Some Active Vision systems use the idea of a *fovea* to reduce computational overhead for real-time tracking systems. In the human eye, the fovea is a very small (less than 1mm diameter) region of the retina in the exact centre of the visual field. It has a very high concentration of visual receptors compared to the rest of the retina and is believed to be of paramount importance in human vision (Nalwa, 1993). To gain information about a new scene, humans' eyes tend to be in constant motion, fixating their attention (by aligning their fovea) on one location for a moment, then switching to the next in rapid succession. Instead of holding still and spending a long time trying to understand the whole of a particular gaze, information about the world is built up gradually by considering only a portion of it at a time.

Foveal computer vision systems act in a similar manner. Instead of spreading the computing resources equally across an entire image, useful portions of the image are identified, and the available computational resources are concentrated towards them. A common kind of foveal system is based on the concept of active contour models, or *snakes* introduced by Kass, Witkin and Terzopoulos (Kass et al., 1987). In their most general form, snakes are energy minimising splines controlled by external constraints. The snake energy functions often are designed so they are minimised when the snake is aligned with a feature, such as a step edge. Edge detection with snakes is reduced to a minimisation problem.

By exploiting inter-frame motion coherence, snakes are allowed to slowly converge over multiple iterations. These iterations may operate on the same image, or multiple images in a video sequence. For each iteration, measurements are restricted to small windows of

interest near the snake, so computational resources are conserved.

The dynamic and adaptive contours of Curwen and Blake (Curwen, 1993; Blake and Isard, 1994), augment snakes with ideas from model-based vision and predictive and adaptive control systems. The shape of the contours is restricted to be one of a pre-defined set of deformations of a user-specified template. A second order predictive model using a Kalman filter (Bar-Shalom and Fortmann, 1988; Grewal and Andrews, 1993) incorporates past contour position and velocity to make the tracking more robust. These predictive models can be learned from training examples (Blake and Isard, 1994; Blake et al., 1995; Baumberg and Hogg, 1995b; Reynard et al., 1996). Better prediction reduces the dependence on measurements. The net result is a further reduction in the resources consumed by the tracking process and improved tracking performance.

One current research area in the uses of active contours is the tracking of cars in video sequences from overhead cameras. Koller, *et. al.* (1994) use image segmentation based on image differencing to initialise affine contours similar to those described by Blake, *et. al.* (1994). Occlusions are detected by tracking the closest objects first. If more distant objects are occluded by nearer ones, the prediction routines of the contours are adjusted so as not to skew the results. The relative distance of the objects is determined by their height on the image, with lower objects being assumed closer to the camera than far ones. The image differencing data is updated based on a weighted median that is updated every frame. This added computation makes their process non real-time.

Ferrier, *et. al.* (Ferrier et al., 1994) also use affine contours for tracking traffic scenes. By learning typical motions through training data, they are able to track vehicles without resorting to image differencing. They maintain that image differencing is not suitable for traffic surveillance because of camera vibrations caused by wind and other traffic. Approximate information about traffic speed is recovered through a simple calibration based on road markers on the side of the road. Only single cars are tracked, but the tracking is done in real time, and they speculate that multiple car tracking is feasible in real time on modest

computational hardware.

Active contours can also be used to recover shape from motion. Blake and Cipolla (Blake and Cipolla, 1990) use active contours to track the apparent contours of objects. By coupling this tracking with known camera movements, the convex hull of the object can be recovered. This recovery process requires known quantitative motion of the camera. The related problem of recovering relative depth of objects in a scene can be solved without quantitative knowledge of the camera movement. Through the use of parallax, robust information about the scene can be recovered. It is possible to discriminate whether features lie on the surface of an object or are part of the occluding contour. This, coupled with the use of the relative depth information allows the system to do real-time visual navigation in scenes that contain curved objects.

2.4 Visual tracking

Visual tracking is the recovery of the position of one or more objects (known as targets) viewed in a sequence of images over time. The two most common kinds of tracking are flow based tracking and feature based tracking.

2.4.1 Flow based tracking

Flow based tracking methods try to estimate image motion in the large through the examination of the apparent motion of individual pixels. Horn and Schunk (Horn and Schunk, 1981) define image flow as the distribution of apparent velocities of brightness patterns in an image. By globally analysing a pair of images from an image sequence they can compute the image flow between this pair of images. The spatio-temporal derivatives are shown to be the image flow under the following assumptions. First, all points present in the first frame must also be present in the second frame, so that no points enter or leave the image. Secondly, the image irradiance or brightness of any point on an object seen in the first frame is assumed to have that same brightness in the second image. Since the aperture

problem states that, based on local information, the velocity of the image flow can only be recovered in one dimension (the direction perpendicular to the image gradient) instead of the two dimensions required to fully recover image velocity, a smoothness constraint is invoked where the magnitude of the image velocity is minimised.

The apparent whole image flow doesn't necessarily relate to the movement of specific objects in the image. Intensity based flow algorithms track the movement of isobrightness contours, which may move independently from the objects they lay on. An obvious, albeit contrived, example of this is of a rotating white disc on a black background (Horn and Schunk, 1981). While the disk is moving in the image, the image flow detects no such movement. The example is contrived because there is no detectable information in the image – any technique to recover motion in the image based on the image data is bound to fail. A more realistic example is the image of a rotating sphere that has a large specular reflection on it. If the sphere has texture on it, the image-flow algorithm will be able track the movement of the surface of the sphere in regions away from the reflection. However, the reflection never moves, so the image-flow algorithm reports no motion in that location, in spite of the fairly obvious motion around the area. A feature based tracker may be able to use other information about the scene to conclude that motion is occurring in spite of the specular reflection, but pixel-based intensity flow algorithms have no underlying geometrical assumptions on which to make such an interpretation.

Intensity based image flow techniques can still be useful for obtaining qualitative information about movement in an image. Murray, *et. al.* (Murray et al., 1994) use intensity-based image flow to help segment movement in images. By implementing subsampling and Gaussian smoothing in parallel on multiple transputers, they are able to estimate the image flow in real time. This image flow is used as a basis for their motion detection and image segmentation procedures.

2.4.2 Feature based tracking

As Nalwa (Nalwa, 1993) points out, the spatiotemporal-differentiation approach to motion-field estimation is limited, since it estimates the motion of isobrightness contours, which may or may not correspond to motion in image sequences. Most work in tracking has been based on the motion of features extracted from images instead of the images themselves.

Corner tracking

The specific kinds of features being tracked varies with the application. The most common kind of feature for tracking is "corner" tracking. Corners are defined to be locations with high contrast in two dimensions. Tracking corners is preferable to tracking lines because corners allow estimates of two degrees of freedom. The aperture effect limits measurements on lines to only provide one degree of freedom.

One corner-based tracker was developed by Shapiro (Shapiro et al., 1992; Shapiro, 1993). Using the corner detector of Wang and Brady (Wang and Brady, 1992), he tracks the movement of corners through image sequences in a way that elegantly handles temporary creation and loss of corners due to noise, occlusion and unocclusion. To reduce the computational complexity of the correspondence problem, small inter-frame movements are assumed. Consequently, only matches between corners near each other in both images are considered. The final corner match is made based on local image patch correlation, and matches must be reciprocal (if corner x in image i "chooses" corner x' in image $i+1$ as its best match, corner x' must also choose corner x as its best match). In the case where no good matches exist for a corner, a "ghost match" is created. The movement of these ghost corners is predicted for a short time in the hope that the feature will reappear.

The DROID tracking system (Harris, 1992a) uses Kalman filters tracking corners in disparity space[1] to recover a 3D model of the world in real-time (5 Hz) once a corner map

[1]Disparity space is a method of using Kalman filters to track features in images. By making the state vector $(x/z, y/z, 1/z)$, the disparity space system is more numerically stable than tracking using Cartesian (x, y, z) coordinates.

of each image has been computed. To generate features for tracking, the Harris/Stephens corner finder (Harris and Stephens, 1988) is applied to each image in the sequence. The corner map is then correlated with the map from the previous image; the resulting information is fed into a Kalman filter to estimate ego-motion, as well as the 3D structure of the surrounding environment.

Other feature tracking

Corners are not the only useful feature for tracking. Dynamic and adaptive contours provide tracking based on information extracted from measurements along contours (Blake and Isard, 1994; Curwen, 1993). These measurements usually correspond to the presence of an intensity edge, though methods that model image backgrounds (Rowe and Blake, 1995; Rowe, 1996) or the appearance of the target (Rowe, 1996; Kaucic et al., 1996) maximise the likelihood that a feature is in a specific location.

Region based trackers track the movement of groups of pixels that share a common characteristic such as colour or texture. Huttenlocher, Noh and Rucklidge (1993) use the partial Hausdorff distance to compare the edge information in a sequence to build 2D models of moving objects and to track the models through complex motions against complex backgrounds.

Combinations of the above methods are also useful. Koller, *et. al.* (Koller et al., 1994) use image differencing to segment images into regions with movement. From this segmentation, regions of motion in the image are developed and dynamic contours are fit onto the regions. Yow and Cipolla (1995) combine intensity models with edge models to robustly locate human faces under varying poses.

2.5 Agrotics and computer vision

The applications of robotics and computer vision in agriculture is a rapidly growing field (Marchant, 1991). Agrotics, or agricultural robotics, has found a wide range of uses, as

have various computer vision techniques applied to agricultural problems.

2.5.1 Agrotics

One of the more famous agrotics projects is Automated Sheep Shearing Project at the University of Western Australia (Trevelyan, 1992). Guided by mechanical rather than visual systems, their *Shear Magic* robot is capable of completely shearing a sheep in 17 minutes from loading to release; in this time, the shear operates only 4 minutes, and they hope to reduce this to 2.5 minutes. A deformable model of a sheep was developed that account for the variability between sheep. By gathering weight and several distance measurements on the sheep and performing a principal component analysis on the data, they were able to estimate the location of any point on the sheep to within 60 mm and the optimal shearing angle to within 6 deg. Beyond that, movement of the shearer was controlled by pressure feedback from the actual head of the robot. As an aside, the author notes that the ultimate success of the robot will be based on politics and economics, not technical aspects.

2.5.2 Computer vision in agriculture

Some recent applications of computer vision in agriculture include various works on fruit grading as well as the harvest of somatic embryos. In his research on detecting defects in apples, Yang notes that differentiating blemishes (i.e. cuts and bruises) from other, non-defective areas (such as stalks and calyces) is generally difficult using 2D information alone. Observing that stalks and calyces tend to occur on concave regions of the surface of the apple and defects tend to occur on the convex regions, he uses structured light to recover information about the surface curvature of the apples. This curvature combined with 2D information such as blemish shape and size is fed into a previously trained neural network. Blemishes are identified with an accuracy of 96.6% in tests.

In another fruit grading application, Singh, *et. al.* graded peaches using colour image analysis in tightly-controlled lighting conditions. A primary consideration in the grading of stonefruit (peaches, apricots, etc.) is its skin colour. A classifier based on a Bayes decision

rule was created from measurements of colour taken by a colorimeter of peaches in their database. On test data, the colour chosen by the classifier agreed with manual classification in 46% of the tests, and selected the correct colour standards out of thirteen in 75% of the tests. The grade of the fruit was determined next. Because the criteria delimiting how to determine the grade of the fruit were explicit, it was easy to use the output of the classifier to estimate the grade of the fruit. However, the grade chosen by the system often disagreed with manual classification. This was thought to be because fruit graders followed a different set of rules than those outlined by the grading criteria. The authors were confident that retraining the system based on decisions made by actual fruit graders would eliminate such discrepancies. They also felt that by using lookup functions for the decision rule, a commercial implementation of their system would be able to grade 5-10 fruit per second, which they considered fast enough to be commercially viable.

Harrel, *et. al.* (Harrel et al., 1993) describe a system capable of identifying and gathering mature somatic chick embryos from a stream of embryos of varying maturity. Embryos were introduced to a narrow (3mm) square glass tube and an image was taken as they passed in front of a video camera along the tube. Various geometric measurements (including length, width, curvature of perimeter, location of centroid, and other measurements) of the occluding contour of the object were used as the input to a three layer neural network, which decided whether the embryo was appropriate for harvesting. After passing in front of the camera, the embryo was tracked by a array of infra-red emitting/detecting pairs. Then if chosen for harvesting, it was ejected from the tube into a gathering chamber. One embryo was analysed every four seconds. The harvester successfully identified 28% of the harvestable embryos, and increased the homogeneity of the population from 56% to 88%.

2.5.3 Combined computer vision/agrotic systems

Combined computer vision and robotic systems are very common as well. Lefebvre, *et. al.* in their project use active vision techniques to take core samples from potatoes for dis-

ease testing (Lefebvre et al., 1993). Potato sprouts are located for sampling by examining 3D reconstructions of the potato. Three dimensional information was obtained through a combination of image differencing, active vision, and shape from shading under controlled lighting conditions. Potatoes were placed on a skewer and photographed against a black background. The high contrast between the potato and background made background suppression easy. The silhouette of the potato was analysed for protuberances, which correspond to sprouts. The potato was then rotated 90° to bring the sprout into a full frontal view, and re-identified by looking for regions of high variance in the image. The measurements from the two images were combined to recover 3D information sufficient for a robot to core the sprout. The core was subsequently analysed for disease. Sprouts were located with a success rates are 93% in a test set of potatoes selected from 6 different species.

In a second method for recognising sprouts, far ($1 - 12\mu$m) infrared images were used. The high surface area and low mass of the sprouts mean that when exposed to hot air they heat up much more quickly than the rest of the potato. The contrast in the subsequent infrared images is sufficient for sprouts to be detected by thresholding. Success rates are similar to the 3D method, and the image analysis is much faster, but practical problems (such as cost and cooling of the camera and test chamber) associated with the use of far infrared cameras still need to be solved.

Automated grasping of potatoes by two-fingered robots has been studied by Taylor, Cox and Blake (Taylor et al., 1994). They use imploding dynamic contours to obtain a B-spline approximation of the occluding contour of a potato from an overhead view. Apparent grasps from this viewpoint are computed through the analysis of the symmetry and anti-symmetry sets of the contour. Grasping points are tracked as the camera servos around the potato in a gradient descent search for better grasps.

Systems capable of automatically recognising partially occluded citrus fruit in a tree canopy under natural lighting conditions have been successfully developed (Plá et al., 1993). The dichromatic reflection model (Klinker et al., 1988) states that the measured colour

emanating from a given point on an object will be a linear combination of two colours – the surface colour and the specular colour. The perceived colour will lay on a plane in (RGB) colour space, and pass through the origin. A plane passing through the origin can be completely described by two angle measurements describing direction. Consequently Plá transforms his RGB colour image information into this angle representation, then makes a cluster-based trainer that makes its decisions in this space. Plá claims completely visible and partially occluded fruits are located 95% of the time.

A similar computer vision system was successfully developed for a robot capable of picking both apples and oranges (Rabatel, 1994). Again a predictive model was used to identify objects in a scene based on their apparent colour in an image. Objects were located in 2D using pixel-based classifiers, and regions of interest were constructed from the classification map. Recognition rates were equal to human location rates. The output of the classification system was then used to guide a robotic arm to pick the fruit. The system was robust enough to work in natural light, but artificial lighting was required for the best performance.

While much work has been done on grading produce of a given kind, VeggieVision (Bolle et al., 1996) works on the object recognition problem as applied to produce. Designed to augment bar-code scanners at food stores, the system can recognise several hundred different kinds of fruits and vegetables even when the produce is in translucent plastic bags. First, polarized light is used to extract the outline of the object without introducing specularities. Then HSI space colour and custom texture histograms are extracted from the image and compared against the database of existing produce. A n nearest-neighbor classification scheme then classifies the object and labels its classification as "sure", "okay" or "unreliable" depending on the support the measurement receives. In the case of "okay" or "unreliable" classifications, the best several matches are presented to the operator, who then completes the classification and allows the system to update itself in a supervised manner. The entire process (except the human interaction) takes approximately one second.

2.6 Criticisms of previous work

While existing research has addressed many difficult problems in recognition of biological entities, many of the solutions are not applicable for the recognition of plants for an automatic weeding system. Many of the systems (Harrel et al., 1993; Lefebvre et al., 1993; Bolle et al., 1996) use controlled lighting situations to aid in the recognition and/or classification stages of the system. Clearly, weeding must be done in the field, so structured lighting systems will not work, and it will be difficult to control the lighting at all. The orange and apple picking robots were capable of classifying in natural light, but there is a large hue difference between the fruit, the leaves of the tree and the ambient sky. While there may be a significant chromatic difference between plantlife and soil, it's not clear that colour alone will be sufficient to distinguish weeds from crop.

The speed of previous systems is also a question. The majority of the systems analyse an entire frame at a time. Rabatel and Plá, *et. al.* claim to be real-time systems, but the rates at which they must analyse the scene are significantly slower than the rates required to segment field images in real-time while a tractor is moving at a reasonable pace. Coupling pixel-based classification schema with region-based systems that operate on only a small region of interest could result in classifiers that work much faster than previous systems.

3

Colour Based Pattern Recognition

One of the primary capabilities of an autonomous robot of any kind is the ability to recognise classes of objects in the world around it. The world of an agricultural robot is a field of crops, and the objects it must recognise are the crops, soil and weeds surrounding it. Such recognition is trivial to a human observer, but the unconstrained poses and diversity of the members of a single class make the problem difficult for a machine vision system. Colour based cues have been found useful for other agricultural robots (Singh et al., 1993; Plá et al., 1993; Rabatel, 1994). The implementation of such colour based pattern recognition systems is complicated, so it is important to understand the theory behind the mechanisms before a lot of time is wasted developing a system that has fundamental flaws.

3.1 Measuring minimal risk for visual data

When developing a recognition or classification system for a given class of objects, the fundamental problem is defining a function $\alpha(\mathbf{x})$, called a *decision rule*, that specifies a map between a given set of measurements $\mathbf{x}$ and a given set of *a actions* or outputs $\alpha_1 \ldots \alpha_a$ (Duda and Hart, 1973). A good decision rule combined with a good choice of measurements

will make a system that is good at classification. If the wrong kind of measurements are chosen as the basis for a classification system, the system is bound to fail.

For example, consider a system designed to examine and recognise different denominations of money based entirely on a measurement of its size. Specifically, assume that the models are £5, £10, £20 and £50 notes, and that the measurements are accurate to ± 1 mm. In this case, it would be quite easy to define a decision rule that was always correct, since all four notes are of sufficiently different size. If, however, the notes were all the same size, (such as \$5, \$10, \$20 and \$50 notes), performance will be no better than random independent of the decision rule chosen, because the measurements do not provide any relevant information about the system. Returning again to the case of the pound notes, if the measurements are more noisy (± 1 cm), it may be possible to construct a decision rule that is correct more often than a random decision rule, but not correct all of the time. Of course, even with meaningful measurements, it is possible to define a poor decision rule.

Through some decision-theory formalisms and Bayes Rule it is possible to quantify this *goodness* of classification and measurement.

3.1.1 Fundamentals of pattern classification

Following the same format as (Duda and Hart, 1973), let $\Omega = \{\omega_1, \ldots, \omega_s\}$ be the finite set of s states of nature (i.e. the true identity of the entity under consideration) and $A = \{\alpha_1, \ldots, \alpha_a\}$ be the finite set of a possible actions for a decision rule $\alpha(\mathbf{x})$[1]. Let the measurement vector $\mathbf{x}$ be a vector-valued random variable and $p(\mathbf{x})$ and $p(\mathbf{x}|\omega_j)$ be its unconditional probability density function (p.d.f.) and conditional p.d.f. respectively. Let $P(\omega_j)$ be the *a priori* probability that nature is in state ω_j. Then through Bayes' Rule (Hogg and Tanis, 1988), the *a posteriori* probability $P(\omega_j|\mathbf{x})$ can be found which describes

[1] In the standard recognition problem the action α_i usually is interpreted to mean "measurement $\mathbf{x}$ is identified as coming from a real-world state ω_j."

the probability that nature is in state ω_j given a measurement $\mathbf{x}$:

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{p(\mathbf{x})} \tag{3.1}$$

To quantify the performance of the decision rule, we introduce the concepts of *risk* and *loss*. Given a state of nature ω_j, a measurement of that state of nature $\mathbf{x}$ and an action α_i, we define the *loss* $\lambda(\alpha_i|\omega_j)$ to be a measurement of the "poorness" of associating action α_i with the given true state ω_j. Then the *conditional risk* $R(\alpha_i|\mathbf{x})$ is the mean loss incurred by choosing α_i for the input $\mathbf{x}$:

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{s} \lambda(\alpha_i|\omega_j)P(\omega_j|\mathbf{x}) \tag{3.2}$$

The *overall risk* is similarly defined to be the mean loss across the entire measurement domain:

$$R = \int R\left(\alpha(\mathbf{x})|\mathbf{x}\right) p(\mathbf{x}) d\mathbf{x} \tag{3.3}$$

A special decision rule, the *Bayes decision rule* α_{Bayes}, is the decision rule defined for a given measurement $\mathbf{x}$ as follows:

$$\alpha_{Bayes}(\mathbf{x}) = \alpha_i \text{ iff } R\left(\alpha_i|\mathbf{x}\right) = \min_k \left(R(\alpha_k|\mathbf{x})\right) \tag{3.4}$$

This minimum risk is called *Bayes risk* and is the best performance that can be achieved (Duda and Hart, 1973)[2] .

3.1.2 Classifier training

Obviously, to choose a good decision rule, *a priori* information is needed about the data to be classified. As more data are analysed, a better decision can be made. Once the data have been gathered the classifier must be trained and evaluated.

[2]It is worth noting that α_{Bayes} may not always be a well-defined function since for a given $\mathbf{x}$ there may be more than one α_i that minimise the risk. However, the method of resolving such ties is not important since all the α_i that minimise the risk locally also minimise the overall risk, thus the resulting decision rule is still a Bayes decision rule.

While the Bayes decision rule is an optimal decision rule, it requires the knowledge of the p.d.f. $p(\mathbf{x})$, the conditional p.d.f. $p(\mathbf{x}|\omega_j)$, the *a priori* probability $P(\omega_j)$, as well as the loss function $\lambda(\alpha_i|\omega_j)$. In practice these functions are usually not known and must be estimated through experimental trials on *training data* and confirmed through further trials on *verification data*. Unlike the data that will be classified once the decision rule has been fully defined, training data have already been classified (i.e. for each measurement $\mathbf{x}$ the state in nature ω_j is known), and consequently estimates of the various p.d.f.'scan be obtained.

Once the p.d.f.'shave been chosen, learned or approximated, the decision rule can be chosen and tested for over-learning on a second set of data, known as the *verification data*, which have also been previously classified. Over-learning is when a recognition system uses recognition criteria that are valid on the training data, but not necessarily on the real data.

As an example of over-learning, consider again the problem of recognising \$5 and \$10 notes based on their size[3]. If the entire training data consists of a single \$5 note (assigned to be class ω_1 and measured to be 119.98mm), and a single \$10 note (assigned to be class ω_2 and measured to be 120.02mm), based on the training data, a best estimate of the *a posteriori* distributions would be

$$p(\omega_1|\mathbf{x}) = \begin{cases} 1 \text{ if } x < 120mm \\ 0 \text{ if } x \geq 120mm \end{cases}$$

$$p(\omega_2|\mathbf{x}) = \begin{cases} 0 \text{ if } x < 120mm \\ 1 \text{ if } x \geq 120mm \end{cases}$$

This is not going to be useful when testing real data, since the decision rule was formulated based on the noise or anomalies in the measurement, as opposed to something that was significantly different between the classes. Re-checking the decision rule with a verification data set helps reduce the risk of decision rules being based on irrelevant information in the measurements. Over-learning can usually be corrected for by using large training sets. However, certain classes of decision rules will be sensitive to over-learning even with

[3]Remember all US paper currency is the same size

good, large training sets, so re-evaluating decision rules through verification data is always important.

3.1.3 Evaluating measurement techniques by finding an upper limit of accuracy

When choosing among various measurement processes for classification, it is important to assess how good each kind of measurement process will be at providing relevant information for a decision rule to act upon. Since formulating decision rules is time-consuming and dependent on the measurement process being used, it is useful to have a way to evaluate the contribution each measurement process gives to the classifier. Through some simplifying assumptions it is possible to obtain a lower bound on the average risk associated with *any* decision rule which depends on a given measurement process.

For simplicity, assume that the actions $A = \alpha_1, \ldots, \alpha_a$ of the classifier are of the form $\alpha_i =$ *"the true state is* ω_i*"* and that the cost function $\lambda(\alpha_i|\omega_j)$ is of the form

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 \text{ if } i = j; \\ 1 \text{ if } i \neq j; \end{cases} \tag{3.5}$$

If the domain of the measurements is discrete, and the number of measurements in a given training set is finite, exact p.d.f.'scan be obtained for a given set of measurements. From those functions a Bayes decision rule can be constructed. By using this Bayes decision rule on the same training data, the minimal risk for the data will be obtained. This risk is minimal for *all* decision rules – it is the best that you can do.

For example, take a data set with states of nature ω_1 and ω_2. From this data, measurements $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ are taken and, continuing the example, the value $\mathbf{m}$ appears five times out of the set of measurements, and that it was obtained once while measuring ω_1 and four times when measuring ω_2. From this information, we can determine that *in this test set* $p(\omega_1|\mathbf{m}) = 0.2$ and $p(\omega_2|\mathbf{m}) = 0.8$. Similarly, $p(\mathbf{m}) = \frac{5}{n}$. Finally, since the conditional risk for measurement $\mathbf{m}$ will be minimised by choosing $\alpha(\mathbf{m}) = \alpha_2$ (i.e. "a measurement of $\mathbf{m}$ means that the object being measured is of class ω_2"), we assign $\alpha_{Bayes}(\mathbf{m}) = \alpha_2$. By

repeating this for the entire range of measurement values, the full $\alpha_{Bayes}(\mathbf{x})$ function can be constructed for the training data.

While the $\alpha_{Bayes}(\mathbf{x})$ function we've just created has minimal risk for the training set, it is most probably not the decision rule with the minimal risk for other data[4]. If the decision rule was constructed as in the previous paragraph, it is only defined over the range of measurements in the training data. In general, this will be a subset of the range of measurements to be encountered in the normal operation of the classifier. However, the Bayes decision rule does provide a metric of the quality of the type of measurements used, since it provides a lower bound on the risk.

3.1.4 Summary

Decision theory provides a mathematical framework for describing pattern classification problems. The Bayes decision rule can be found from the training sets of certain classes of measurement processes. It provides a lower bound on expected risk for any decision rules based on a given measurement process, which can be used to evaluate the utility of such a process for a given pattern classification problem.

A more intuitive measure, *accuracy*, is defined to be one minus the average risk. So, in the case of perfect classification (i.e. for each measurement $\mathbf{x}_j$ taken of state ω_i, then $\alpha(\mathbf{x}_j) = \alpha_i$) the average risk is zero, but the accuracy will be 1. Often accuracy is represented by a percentage, such as 100% rather than a fraction, such as 1.00. For consistency, all further classification metrics will be accuracy-based rather than risk-based, and rates will be written as percentages.

3.2 Different colour space representations for visual data representation

For the classification of pixel information in digital images, the individual pixels are the measurements. The states that they measure are the colour and brightness of a ray of light

[4]The larger the training set, and the more representative it is, the better decision rule $\alpha_{Bayes}(\mathbf{x})$ will be.

entering the camera. Colour can be represented in many different ways. Each representation has its specific advantages and disadvantages. In the previous section, a theory was developed that provides a way to compare the discerning abilities of various measurement techniques. By applying it, it is possible to compare different ways of measuring pixel colour in digitised images, and evaluate how appropriate each is for a given task.

Visible light is electromagnetic radiation with a wavelength of 400 to 700nm, which corresponds to the colours of the rainbow starting with violet and progressing to red (Foley et al., 1990). "Near" infra red light is the "colour" of light just below the visible range (i.e. 400nm). Most CCD cameras are sensitive to this wavelength of light unless they have a special IR filter on them. A spectral energy distribution $P(\lambda)$ represents the energy level at each wavelength.

Humans perceive colour through visual receptors in the retina of the eye known as cones (Nevatia, 1982; Nalwa, 1993) that respond to incoming light distributions by integrating them in a way that emphasises a band of the visible light spectrum. Individual cones are tuned to respond either to red, green or blue light. Though the eye measures light in terms of its intensity in each of the red green and blue bands, humans psychologically perceive colour in terms of a different three measurements (Nevatia, 1982; Foley et al., 1990). *Hue* describes the colour of the rainbow the light is perceived to be most like, for example red, green, or violet. *Saturation* describes how far a colour is from a grey of equal intensity. Greys are completely unsaturated, pastel colours are slightly saturated, while vivid colours are quite saturated. *Lightness* and *Brightness* respectively describe the perceived achromatic intensity of reflecting and self-luminous (i.e. light-emitting) objects.

Computers also usually measure the colour of incoming light by weighted integrals of the spectrum instead of measuring the complete spectrum of the light. Since the human visual system uses three measurements to represent light computers usually do as well (Foley et al., 1990). The qualities of the colours the three measurements measure varies. Different measurements have various advantages and disadvantages that should be considered when

choosing which representation is best for a decision rule.

3.2.1 Visible light, black and white "colour" space

The simplest and most common representation of digital images is a monochrome representation, where a single value represents an integral weighted approximately evenly across the entire visible spectrum of light. The weight is such that the measurement approximates the intensity of light that a human would perceive at that spot.

The most commonly used representation of perceived light for physiological, mathematical and practical reasons is monochrome. Humans demonstrate the ability to effectively interpret a wide range of images without the use of colour. Psychophysical evidence suggests that colour information is treated at a high reasoning level, and that in general it is not used for such low-level understanding functions as motion and edge detection (Ramachandran, 1990), indeed in many people colour-blindness may even go unnoticed.

From a mathematical point of view, many of the basic operations of computer vision, such as edge detection (Marr and Hildreth, 1980; Canny, 1986) and corner detection (Wang and Brady, 1992) have intuitive meanings in black and white images, but their meanings are not often clear in colour images. Often little or no new information is gained from the colour, and psychophysical research has shown that humans have a hard time perceiving edges that are colour only . In an early paper by Nevatia (Nevatia, 1977), after implementing colour edge detection, he noted there was a strong correspondence between colour edges and regular, intensity-based edges.

From a practical standpoint, monochrome images are often preferred because black and white cameras are less expensive, more reliable, work under more diverse lighting conditions and provide cleaner data than colour cameras. Since measurements are scalars instead of vector triples, the bandwidth and storage requirements required for black and white images are one-third that of colour images of equal resolution.

To increase the contrast in the image, it is possible to restrict the bandwidth of the

light reaching the lens by placing one or more filters in front of the lens. This has been used for many applications, most interestingly to read the Dead Sea Scrolls. One band of particular use in the field of plant tracking is the near IR band. Because plants have a very high reflectance in this wavelength, the plants appear much brighter than the soil in such images.

Using restricted wavelength monochrome images is advantageous because the images have improved contrast but still have scalar pixels, thus we can still use the standard image manipulation routines that require scalar pixels.

In spite of compelling reasons for the use of monochrome images, colour information is not without its use. Image understanding can be greatly enhanced by the presence of colour, especially in images of low (luminance) contrast. Clearly, an equiluminant red square on an equiluminant green background can not be perceived without colour information. There is also evidence that colour perception helps contribute to long range tracking skills, as well as other visual processes, (Ramachandran, 1990). Interestingly, this coupled with the observation that humans have difficulty detecting colour edges seems to imply humans don't rely exclusively on edges when tracking.

3.2.2 RGB colour space

A popular method to encode colour information is the Red Green Blue (RGB) method of colour representation. In a RGB representation, each image pixel is represented by a triple (r, g, b) where r is a weighted integral of the spectral distribution that emphasises the red portion of the spectrum, g, the green portion, and b, the blue portion. Normally these integrations are done directly by the red, green and blue receptors of the colour video camera.

A problem with RGB representation of colour is that the relation between the (r, g, b) vector and the perceptual categories of hue, saturation and lightness/brightness is not intuitive (see equations (3.6), (3.7) and (3.8) for an approximate relationship).

3.2.3 HSV colour space

Hue, Saturation and Value (HSV) is a common representation of coloured light in digital images. The values in the HSV measurement vector triple (h, s, v) intuitively relate well to perceptual categories of hue, saturation and lightness/brightness. Value (v) measures the overall brightness of the pixel. Hue (h) is an angle describing the "rainbow" or "pure" colour nearest to the pixel, where red is defined to be the colour at 0°, green is at 120°, and blue is at 240°. Saturation (s) relates how "tinted" the colour is – the ratio of the pure colour to the white light. Equations (3.6), (3.7) and (3.8) (from (Foley et al., 1990)) describe the exact transformation. An equivalent way to think about hue and saturation is that any colour in RGB space can be produced by the sum of a pure colour (the hue) and white light (light with $r = g = b$). Saturation represents the amount the pure colour contributes to the sum. Table 3.1 shows a few example colours in RGB space mapped to their equivalent representation in HSV space.

While intuitive, HSV space is not without its problems. Since the native format of the data from almost all video cameras is RGB, a non-linear transformation must be applied to each pixel to change it from RGB to HSV. Most operations on digital images are done with integer arithmetic, and the image data, already quantised in RGB space, is usually quantised again after being transformed to HSV space. Both information and precision are lost in the conversion process. This, coupled with the singularity in the transformation about $s = 0$, can make h meaningless for significant portions of the image (since the colours of many real world images are relatively unsaturated). Different white balance settings (see section 4.4.2 and (Langford, 1984)) of the video camera can also move the location of this singularity around, causing further problems. (see figure 4.7)

$$v = \max(r, g, b) \tag{3.6}$$

$$s = \begin{cases} (\max(r,g,b) - \min(r,g,b))/\max(r,g,b) & \text{if } \max(r,g,b) \neq 0 \\ 0 & \text{if } \max(r,g,b) = 0 \end{cases} \tag{3.7}$$

$$h = \begin{cases} undefined & \text{if s} = 0 \\ 60°(g-b)/(r - \min(g,b)) & \text{if } r = \max(r,g,b) \\ 120° + 60°(b-r)/(g - \min(b,r)) & \text{if } g = \max(r,g,b) \\ 240° + 60°(r-g)/(b - \min(r,g)) & \text{if } b = \max(r,g,b) \end{cases} \tag{3.8}$$

3.2.4 Other representations of colours in digital images

Although RGB and HSV representations are the two most common representations of colour in digital images, Plá, *et. al.* (Plá et al., 1993) cite other authors' works describing image segmentation and processing based on $L*a*b$ (Luminosity, Green-Red, Blue-Yellow) space (Tominaga, 1990), and IHS (Intensity, Hue and Saturation) space, as well as more ad-hoc descriptions of colour reminiscent of those developed in (Nevatia, 1977). Plá cites further work in the applications of colour image processing to robotic citrus harvesting using IHS space (Cerveron et al., 1990), as well as the use of controlled lighting and narrow bandwidth colour filters for the same reason (Moltó et al., 1992).

To narrow the focus of this research, only visible light black and white, RGB and HSV images will be studied.

3.2.5 Summary

While there are many different and useful ways to represent the colour in an image (including not representing it at all), in the general case no one method has a complete advantage over any other. To intelligently choose which representation is the most useful, a theoretical framework is needed that describes the relationship between the colour of the light source, the surface of an object and its perceived colour in an image. This framework can help guide the choice of one colour representation over another.

colour	name	r	g	b	h	s	v
	red	1.0	0.0	0.0	0°	1.0	1.0
	yellow	1.0	1.0	0.0	60°	1.0	1.0
	green	0.0	1.0	0.0	120°	1.0	1.0
	cyan	0.0	1.0	1.0	180°	1.0	1.0
	blue	0.0	0.0	1.0	240°	1.0	1.0
	magenta	1.0	0.0	1.0	300°	1.0	1.0
	black	0.0	0.0	0.0	*undefined*	0.0	0.0
	grey	0.5	0.5	0.5	*undefined*	0.0	0.5
	white	1.0	1.0	1.0	*undefined*	0.0	1.0
	pink	1.0	0.75	0.75	0°	0.75	1.0
	dark red	0.5	0.0	0.0	0°	1.0	0.5

Table 3.1: **Examples of conversions between RGB space and HSV space** *In HSV space, hue and saturation are invariant to the brightness of the colour. Since all grey points are equidistant from all completely saturated colours, their hue is undefined.*

3.3 The Dichromatic Reflection Model

The Dichromatic Reflection Model (DRM) (Shafer, 1985) has been shown to be useful for colour image understanding (Klinker et al., 1988). In (Plá et al., 1993), Plá, *et. al.* use a colour-based recognition system that is an application of the dichromatic reflection model to harvest oranges from orange trees. They were able to successfully locate 95 percent of the visible fruits, with a 6 percent failure rate.

3.3.1 A mathematical description of the DRM

Image intensity $L(\lambda, i, e, g)$ corresponding to a point on a object is modelled to be proportional to the light $L_s(\lambda, i, e, g)$ reflected at the material surface (or interface) plus the light $L_b(\lambda, i, e, g)$ reflected at the material body. Here λ is the wavelength of the light, i is the angle of incidence (the angle between the surface normal and the incoming light wave), e the angle of exitance (the angle between the surface normal and the viewer), and g is the phase angle (the angle between the incoming light wave and the viewer). See figure 3.1 for a pictorial view of this.

The DRM is based upon three assumptions. Firstly, the surface is assumed to be opaque, inhomogeneous (it has multiple layers), not optically active (it does not fluoresce) and have one significant interface (the interface reflections must only come from one layer of the surface). Secondly, the reflection from the surface is assumed to be isotropic with respect to rotation about the surface normal (the surface isn't polarised). Thirdly, the lighting for the entire scene is assumed to be from a single point source, and inter-reflection between surfaces is assumed to be negligible.

Since the surface is isotropic and of constant colour, it can be assumed that the object surface has the same reflectance spectrum $L_s(\lambda, i, e, g)$ at all points on the surface. This separates the effects of the specular reflection from the effects of the geometric orientation

of the system, which can be modelled by letting

$$L_s(\lambda, i, e, g) = c_s(\lambda) m_s(i, e, g) \tag{3.9}$$

where $c_s(\lambda)$ is the spectral power distribution of the object surface's reflectivity, and $m_s(i, e, g)$ is the geometric scale factor. The same assumptions can be made for the body reflection component L_b, giving

$$L(\lambda, i, e, g) = c_s(\lambda) m_s(i, e, g) + c_b(\lambda) m_b(i, e, g) \tag{3.10}$$

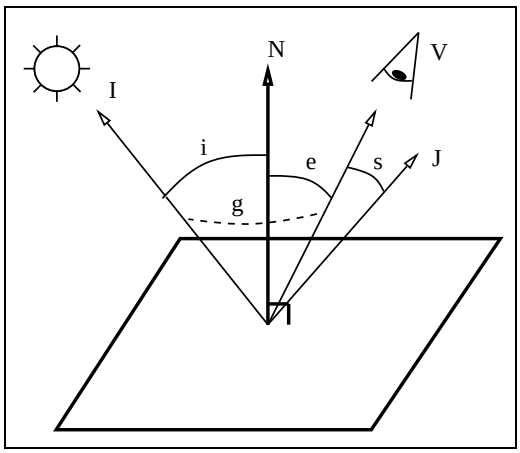


Figure 3.1: **Geometric angles used in the dichromatic reflection model** *The DRM states that the colour emanating from a surface of constant colour depends on properties of the surface and the geometry of the surface normal, viewer and light source. This geometry can be described by three significant angles. The incidence angle i is the angle between the light source and the surface normal. The angle of exitance e is the angle between the surface normal and the viewer's position. The phase angle g is the angle between the viewer's position and the light source. A fourth angle s is significant for predicting the specularity of a point. The ray J is the reflection of the incident ray I from the normal N. When s, the angle between the reflected light and the viewer, is small, the specular component of the light will be high.*

In the case when RGB values are being used, the continuous spectrum of light is reduced to three weighted measurements and the equation can be simplified to

$$\mathbf{L}(i, e, g) = \mathbf{c}_s m_s(i, e, g) + \mathbf{c}_b m_b(i, e, g) \tag{3.11}$$

Where $\mathbf{L}$, $\mathbf{c}_s$ and $\mathbf{c}_b$ are all vector-valued functions of the form (r, g, b).

3.3.2 The dichromatic reflection model for image understanding

While the DRM has some questionable assumptions (most notably that the surfaces are isotropic and that inter-reflection is neglible), the DRM can be used to classify objects in an image based on their colour (Klinker et al., 1988; Plá et al., 1993). When trying to analyse an image, information about i, e and g is not known. If i, e and g are unknown, equation (3.11) can be rewritten $\mathbf{L} = \mathbf{c}_s m_s + \mathbf{c}_b m_b$, where m_s and m_b are now parameters that are allowed to vary in the range $[0, \infty]$. Thus given a constant colour light, the perceived colour of any point on the object must lie on a segment of a plane in colour-space (in this case RGB colour-space) that intersects the origin and is bounded by the rays in the direction $\mathbf{c}_s$ and $\mathbf{c}_b$. For simplicity, this segment will simply be referred to as a planar segment.

By observing the colour of points coming from a known object ω, the $\mathbf{c}_s$ and $\mathbf{c}_b$ terms can be recovered. A database of $\mathbf{c}_s$ and $\mathbf{c}_b$ vectors can be constructed. In future observations, the distance between a perceived colour of light $\mathbf{L}$ and planes specified by the vectors in the database can be computed, and the source of the light can then be identified as the object with the nearest plane. This process can be further simplified by assuming the colour of specular reflections is the colour of the light source itself ($\mathbf{c}_s$ has the spectral energy distribution $\mathbf{P}$). Thus the graph of the perceived colours of n objects in RGB space will be a set of n planar segments, all intersecting the origin, and all sharing the ray specified by $\mathbf{P}$. An object can be identified based on its measured colour simply by determining which blade of the pinwheel the colour point lies closest to (see figure 3.2).

3.3.3 Implications of the dichromatic reflection model on the choice of colour representation

The DRM states that light perceived as emanating from a uniformly coloured object will lie on a planar segment in RGB space intersecting the origin. When alternative colour representations are considered, it is important to choose a representation that groups points on this planar segment together. Obviously RGB space does preserve the grouping of similar

points, since the theorem is specified in RGB space.

HSV space requires a bit more analysis. From equation (3.6), it is easy to see v gives no relevant information for the distance between a point of colour and one of the planar segments in RGB space. In the general case, h and s give information of limited use. A planar segment bounded by $\mathbf{c}_s$ and $\mathbf{c}_b$ will map onto a range of hues and saturations. Both saturation and hue are invariant to changes in intensity (i.e. $\text{hue}(c\mathbf{x}) = \text{hue}(\mathbf{x})$ and $\text{saturation}(c\mathbf{x}) = \text{saturation}(\mathbf{x})$). The upper limit of the saturation will be either the saturation of $\mathbf{c}_s$ or $\mathbf{c}_b$. To evaluate the lower limit of the saturation, the entire line $x\mathbf{c}_s + (1-x)\mathbf{c}_b$ must be considered. The range of hues will be a single band of values in hue space not greater than 180° bounded by $\text{hue}(\mathbf{c}_s)$ and $\text{hue}(\mathbf{c}_b)$.

Under the assumption that the colour of specularities is the colour of the light source ($\mathbf{P}$), it is generally a bad idea to classify more than 2 classes of objects based on hue. Since all n classes of objects will be specified as ranges of hue between $\text{hue}(\mathbf{P})$ and some other hue, the different ranges will have to overlap (see figure 3.2).

In one case, also illustrated in figure 3.2, the DRM indicates that classification based on hue will be a very good idea. In the case where $\mathbf{P}$ is on the line $r = g = b$, the hue of $\mathbf{P}$ is undefined. Consequently, all points on the planar segment bounded by $\mathbf{P}$ and $\mathbf{c}_b$ (except those on the ray $\mathbf{P}$ itself) will have the same hue. While this is a "degenerate" case, it is one that often occurs. When a video camera is "white balanced", it is adjusted so the colour of the ambient light lies on the line $r = g = b$. This is done to prevent the image from acquiring an overall tint. Since the colour of the ambient light source is the colour of the specularities, a camera that is properly white-balanced will produce images that contain objects that can be identified through discrimination based on hue alone.

3.3.4 Conclusion

The DRM describes the relation between the reflection characteristics of objects and the colour of light perceived reflecting from them. It also can be used as a way to identify

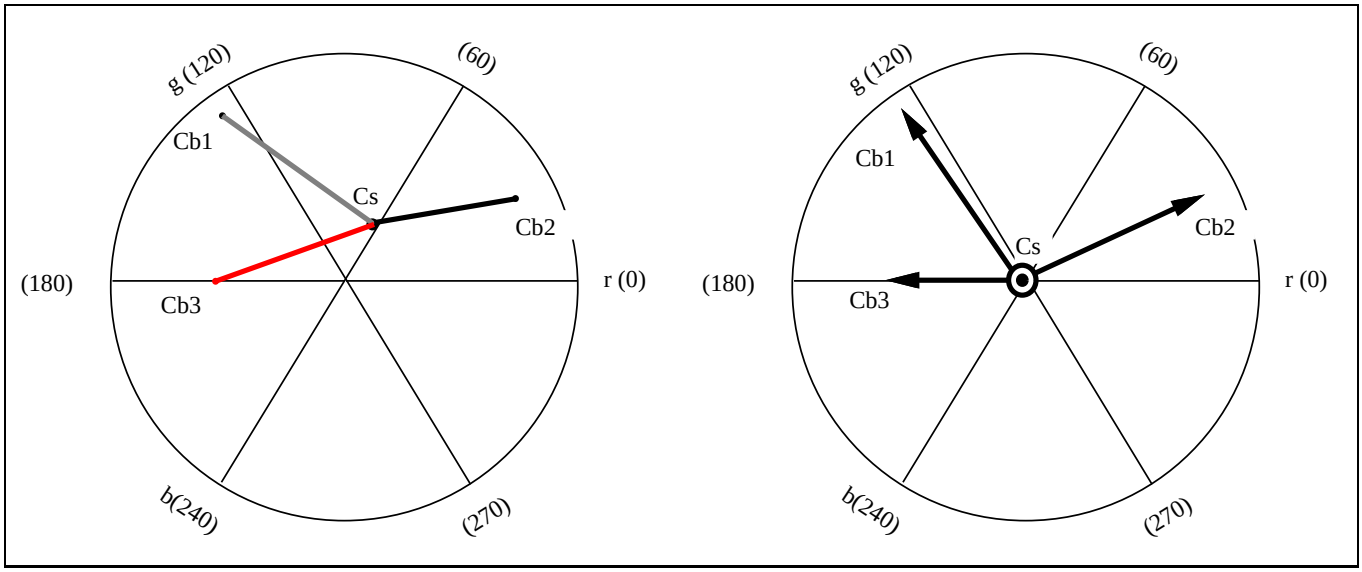


Figure 3.2: **Illustration of the dichromatic reflection model in RGB space and HSV space.** *The DRM predicts that the perceived colour of light emanating from a surface of a constant colour will be a linear combination of two vectors: the colour of light reflected from the surface (specular reflection) of the object, and the colour of light reflected from the body (diffuse reflection) of the object. Assuming that the colour of the specular reflections is the same as the colour of the light source (thus identical for all objects), the plane segments of differently coloured objects will have a single ray in common that is the ray describing the colour of the light source. If the ray is not aligned with the line $r = g = b$, the hue of all points on a given plane segment will be a range of values bounded by the hue of $\mathbf{c}_b$ and $\mathbf{c}_s$. Classification based on hue alone will be ambiguous (for example, on the figure on the left some points from the plane segments specified $\mathbf{c}_{b1}$ and $\mathbf{c}_{b3}$ have a hue of $120°$). In the case where $\mathbf{c}_s$ is aligned with $r = g = b$, the plane segments map onto single rays in hue space, so no ambiguities remain and classification will improve.*

classes of objects based on their colour. Two representations of colour, RGB and HSV, can provide useful measurements for distinguishing objects based on this theory. In RGB space, colours perceived as emanating from a singly coloured object lie on a planar segment passing through the origin and a ray describing the colour of the ambient light source. In HSV space, under proper white-balancing of the camera, the perceived hue of colours of singly-coloured objects will be a constant. This is very sensitive to the white-balancing, and may or may not be useful in practice. Testing is necessary to evaluate this (see chapter 4).

3.4 Summary

By developing methods to compare the suitability of different kinds of measurements for classification processes, we have provided a way to judge which representations of visual light are most appropriate for use as a measurement in that process. By studying a theory relating the colour measured in the digitised image and the "true" colour of the object in the image, we have a better idea of what to search for when identifying an object by its colour. The next step is to actually run experiments on the training data and measure which representations are useful and which are merely theoretically interesting.

4

Applications of colour based pattern recognition

A series of techniques have been developed allowing the comparison of different measurement techniques for pixel based classification. These techniques are now applied to a real problem, in this case, the classification of plant life in digital images of cauliflower fields. The upper bound of the accuracy of different measurement techniques is found, and a few simple pixel-based classifiers are constructed and compared in terms of their overall accuracy.

4.1 Data preparation

Eleven sequences of images of cauliflower fields were chosen for study from a database of agricultural image sequences. The sequences were digital photographs of cauliflowers aged from one to three and a half months taken with a video camera under both clear and cloudy skies (see table 4.1). From these eleven sequences, the first image of each sequence was chosen for analysis. In addition, four images from sequence 1 were chosen (see figure 4.1 for sample images from each of the sequences).

A set of four copies of each chosen image were made. A paint program was used to

Image	Camera	Angle	Height	Weather	Filters	Age
1	B	90°	177cm	sunny	polarizing	2.5 months
2	B	90°	177cm	sunny	polarizing	2.5 months
3	B	90°	176cm	cloudy	none	3.5 months
4	B	90°	176cm	cloudy	none	3.5 months
5	B	55°	176cm	cloudy	none	3.5 months
6	B	40°	90cm	cloudy	none	2 months
7	B	20°	90cm	cloudy	none	2 months
8	B	20°	90cm	cloudy	none	2 months
9	A	35°	90cm	sunny	none	1 month
10	A	35°	90cm	cloudy	none	1 month
11	B	40°	90cm	sunny	none	1.5 months

Table 4.1: **Camera information for images in figure 4.1** *In the table, angle is measured from horizontal, and height is measured from the ground. Camera A was a Panasonic WV-BL200/WV-BL204 with a resolution of 280 (H) x 350 (V) and a 2:1 interlace. Camera B was a Sony XC-711/711P with a resolution of 768 (H) x 493 (V) and a 2:1 interlace. Note that Images 1 and 2 were taken on a sunny day, but were shaded from direct sunlight.*

blacken out all pixels not recognisable as cauliflowers in the first copy. With the second copy, all pixels not recognisable as soil were blackened, and in the third, all pixels not recognisable as weeds were blackened. The fourth copy was left unmodified as a reference. The partially blackened copies were then combined into a single "map" image. Each pixel in the map had a value labelling it either as being cauliflower, soil, weed or unclassified. The few pixels that were erroneously classified as belonging to more than one class were marked as unclassified.

4.2 Classification of field cauliflowers in monochrome images

When working with individual black and white images from the fields, it quickly became clear that the visible light black and white images were not sufficient for identifying weeds. In figure 4.2, a black and white reproduction of image 1, no weeds are visible. In a map created by a person looking at a colour image, weeds are shown to cover a significant portion of the image. This improved perception in the presence of colour was typical throughout the test images. Since visual perception in machines is usually inferior to visual perception in humans, the chances that a machine will be able to discern the different classes of plants

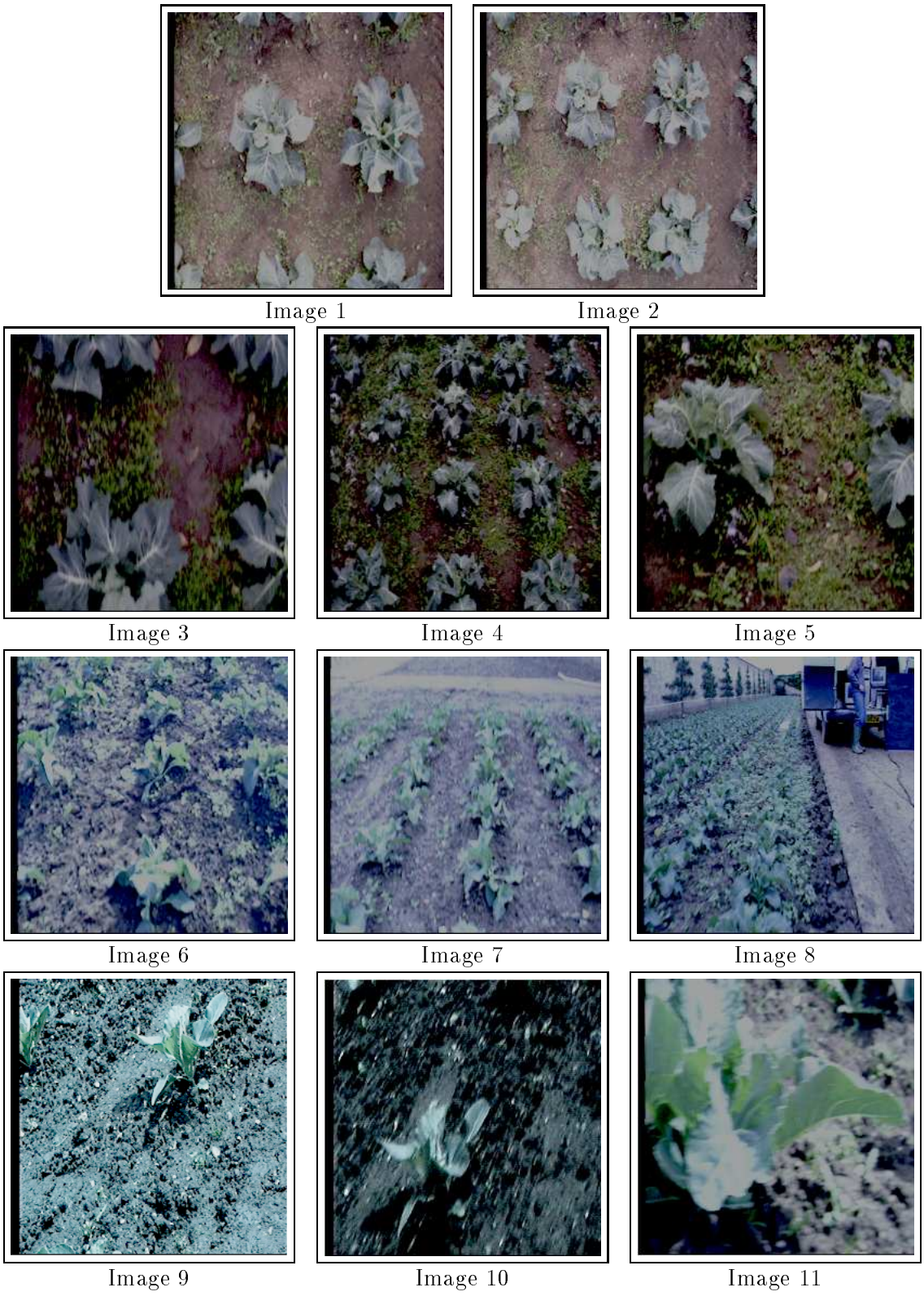
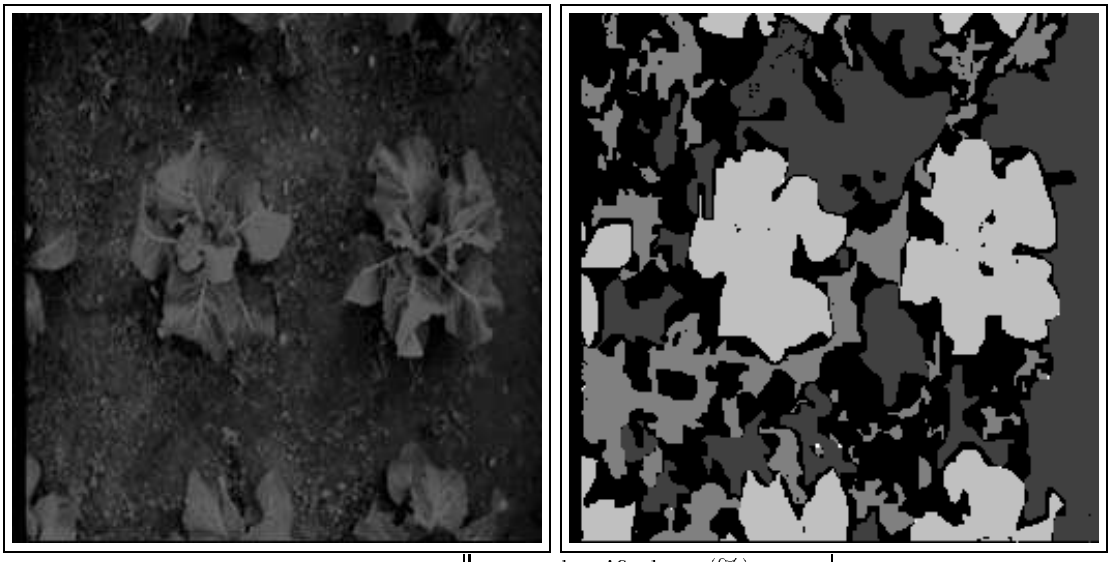


Figure 4.1: **A catalogue of the images used** *For the details of the conditions the images were taken in, see table 4.1*



	classified as (%)		
region type	cauliflower	soil	weed
cauliflower	73.93	10.89	21.99
soil	26.02	89.11	78.01
weed	0.00	0.00	0.00

Figure 4.2: **No good decision rule exists for classifying visible light monochrome images of cauliflowers** *The original field image (image 1 on left) converted to black and white, and a map image on the right showing a hand-labelling of the original colour image. The greys in the map signify, from lightest to darkest, cauliflowers, weeds, soil, and unclassified pixels. The weeds in the monochrome image on the left are barely visible. Using a Bayes decision rule, all pixels corresponding to weeds are misclassified, and only 74% of the cauliflowers and 89% of the weeds were correctly classified. The Bayes decision rule, which is optimal, classified this image better than the rest of the monochrome images in terms of the total number of pixels misclassified. Based on this evidence, it was concluded that visible light monochrome images provided insufficient information to classify field scenes such as this.*

in the test images is small.

A Bayes decision rule classifier was created for each image, and its accuracy was computed. On all the images, classification accuracy was poor. On image 1, the test image with the highest accuracy, none of the weed pixels in the image were found, 74% of the cauliflower pixels and 89% of the soil pixels were found. The measurement process was concluded to be inappropriate for the classification problem, and was removed from further consideration.

4.3 Classification of field cauliflowers using RGB space

Visible light black and white classification proved to be difficult on the images from the test sets. Even for humans, weeds in the monochrome images were difficult to discern. However, in full colour images the three classes of objects (soil, cauliflowers and weeds) were clearly visible to the human eye. This implies that there is information in the colour of the image that can help recognition.

The native representation of colour on most computer systems is RGB. Individual applications may choose to represent the information in other ways that are more intuitive or convenient. Since they are merely transformations of RGB measurements, they can not increase the amount of information the measurement contains. Therefore, if *any* representation of colour is capable of classifying the scenes, then at least RGB will be.

4.3.1 Upper limit of accuracy

In order to get an upper limit on the accuracy possible for single pixel classification using colour data, a program was written to compute the Bayes decision rule for RGB space. To do so, the program did a full resolution ($1 \times 1 \times 1$ pixel) histogram for each of the three classes.

In addition to specifying the Bayes decision rule for a given set of data, a histogram contains sufficient information to compute an upper bound of the accuracy. For example, if there were ten pixels with value $(105, 72, 98)$ known to be cauliflowers, five of that value that are soil and two that are weeds, the Bayes decision rule would label all 17 as cauliflowers. Ten would be classified correctly and seven incorrectly. By counting correct and incorrect labellings, it is possible to compute the accuracy for the Bayes classifier. The Bayes decision rule is optimal, so this accuracy is an upper bound for all single-pixel classifiers in RGB space. Since the data are measured in RGB space, no additional information is obtained by analysing the data in another colour space. Thus the Bayes rule is an upper bound for all single pixel classifiers on the test set.

The first column of table 4.3 shows the performance of the Bayes classifier on the test images. The accuracy of the classifier was above 95% for most of the colour images and occasionally reached 99%. In the worst case (image 2 – see figure 4.1) an accuracy of 85% was found (see table 4.3). This shows there exist accurate single-pixel classification functions for colour images. The problem then becomes to find practical and robust classification functions.

4.3.2 A simple linear classifier in RGB space

Duda and Hart (Duda and Hart, 1973) develop optimal linear classifiers for data sets characterised by certain Gaussian distributions. As a test, the statistical moments of the test data were computed to see if any of the optimal linear classifiers mentioned were appropriate.

For data characterised by Gaussian *a priori* distributions, there is one general case when it is possible to obtain an (optimal) Bayes decision rule that is a linear classifier (Duda and Hart, 1973). Define μ_i to be the mean of the measurements of class ω_i, and let Σ_i be the covariance matrix of the measurements of class ω_i. When the covariance matrices of all the classes are identical (the covariance matrices of the classes are of the form $\Sigma_i = \Sigma$), and the *a priori* probability of all the classes $P(\omega_i)$ are equal, The Bayes decision rule is

$$\alpha_{Bayes}(\mathbf{x}) = \alpha_i \text{ if } D(\mathbf{x}, \mu_i) = \min_j D(\mathbf{x}, \mu_j)$$

where

$$D(\mathbf{x}, \mu_i) = (\mathbf{x} - \mu_i)^t \Sigma^{-1} (\mathbf{x} - \mu_i)$$

which is the squared Mahalanobis distance from $\mathbf{x}$ to μ. When the priors are not equal, the rule becomes

$$\alpha_{Bayes}(\mathbf{x}_i) = \alpha_i \text{ if } g_i(\mathbf{x}_i) = \min_j g_i(\mathbf{x}_i)$$

where

$$g_i(\mathbf{x}_i) = -\frac{1}{2} D(\mathbf{x}, \mu_i) + \ln P(\omega_i)$$

The quadratic term of the Mahalanobis distance $\mathbf{x}^t\Sigma^{-1}\mathbf{x}$ is independent of i and can be dropped, leaving a linear discriminant function.

When the covariance matrices are of the form $\Sigma_i = \sigma^2 I$ (i.e. the measurements of all the classes are all independent of one another, and they all have the same variance) the Mahalanobis distance is merely a constant times the Euclidean distance. Geometrically, the samples fall in equal-size hyperspherical clusters, with the samples of class ω_i being centred around the mean μ_i. A measurement $\mathbf{x}_i$ is classified as being a member of the class with the nearest mean μ_i. The Mahalanobis distance in the more complicated case of $\Sigma_i = \Sigma$, can be interpreted in the same way, provided a linear transformation is applied first to the entire system (means, measurements and covariances) so that in the new basis, the covariance matrices satisfy the $\Sigma_i = \sigma^2 I$ constraint. As before, the quadratic term of the distance can be dropped, and the discriminant function is linear.

Statistical analysis of data

To see if the minimum-Mahalanobis distance classifier was appropriate for the data in the cauliflower database, the first and second order moments of the data were computed (see table 4.2). As the distribution functions of the data were unknown, they were modelled by a Gaussian distribution, as in Crisman (Crisman, 1992). Projections of several of the distributions are shown in figure 4.3. While the data are clearly not Gaussian (see the spurious tails on the distributions of image 7), the distributions are unimodal, ellipsoidal and roughly symmetric, so the Gaussian distribution is a reasonable first approximation.

The high values in the off-diagonals of the covariance matrices (see table 4.2) show that the data is poorly approximated by the $\Sigma_i = \sigma^2 I$ case. Comparing the statistic moments of the measurements of each image studied revealed that the covariance matrices of the measurements of the three classes were similar but not identical. The ratio of $[\Sigma_i]_{(x,y)}$ to $[\Sigma_j]_{(x,y)}$ (where $[\Sigma_i]_{(x,y)}$ refers to the element of Σ_i in the xth row and yth column) was as small as 1.1 in some cases, but in other cases, it was as large as 1.4. Based on this

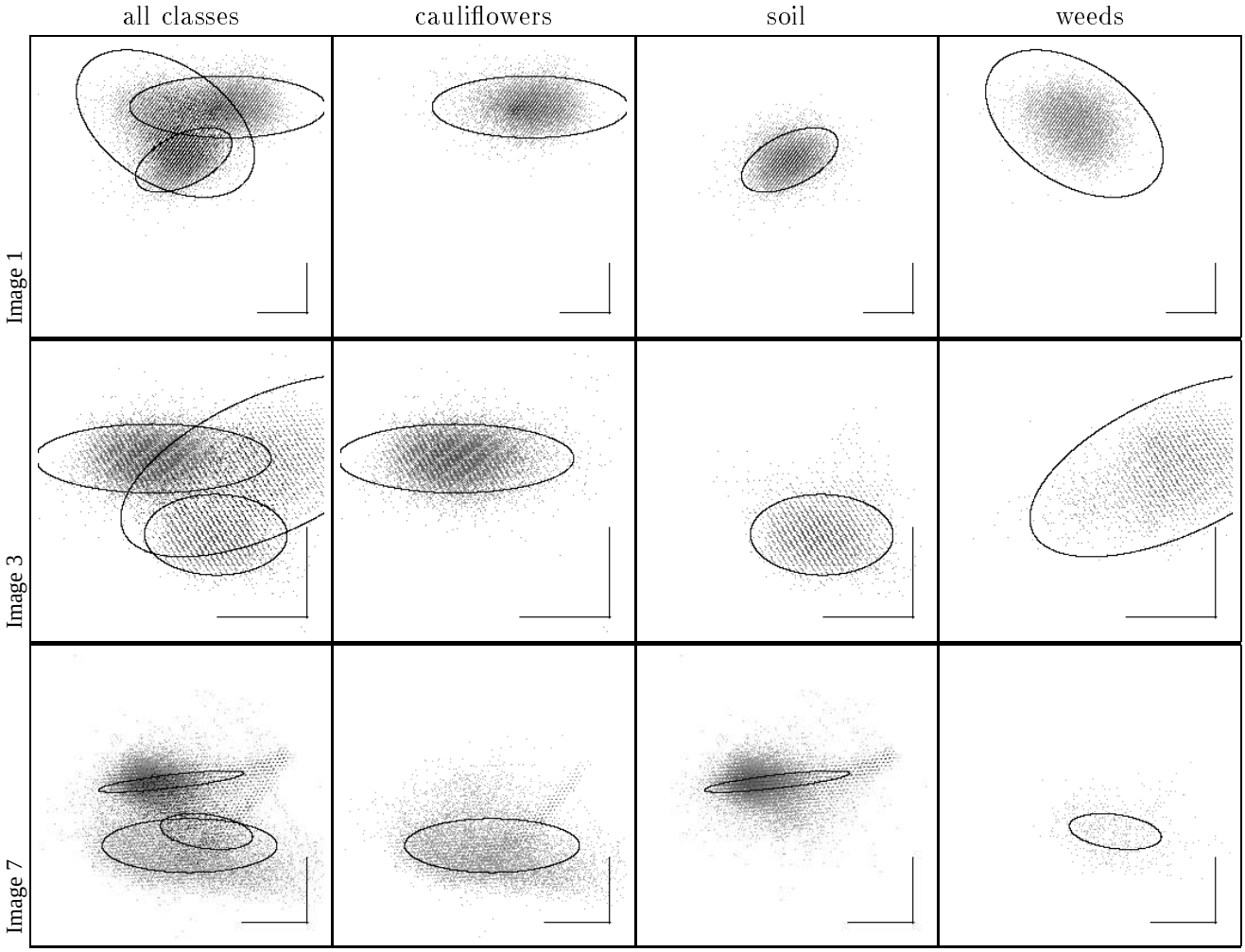


Figure 4.3: **Linear Separability of RGB data** *These pictures show 2D projections of the RGB space of images 1,4 and 7. Given a class covariance matrix with eigenvalues* $\lambda_0 > \lambda_1 > \lambda_2$ *corresponding to eigenvectors* Λ_0*,* Λ_1 *and* Λ_2*, points in RGB space are projected down onto the plane specified by* Λ_1 *and* Λ_2*, thus reducing the dimensionality of the classification problem. Images in the first column show graphs of the combined data of all three classes. The second, third and fourth columns of images shows the sets of measurements identified as cauliflowers, soil and weeds respectively. The data points are moderately, but not completely separated. When* Λ_1 *or* Λ_0 *were removed the data points were poorly separated. These graphs illustrate that a pixel-based classifier will never achieve very high performance because the data of different classes are not completely separated from each other even in the best case. Ellipses of constant covariance are drawn around each class mean. The graphs have been scaled and cropped to show only relevant detail, and a scale has been included that shows* $\frac{1}{20}$ *of the range of the original data.*

observation, the use of the decision rule for the $\Sigma_i = \Sigma$ case was also ruled out.

The dominant eigenvalue of the covariance matrix in all the test images was found to be significantly larger than the other eigenvalues, and it always corresponded to an eigenvector

class	λ_0	Λ_0	λ_1	Λ_1
cauliflower	0.021	$(-0.61, -0.56, -0.56)$	0.00049	$(-0.47, -0.32, 0.82)$
soil	0.0096	$(-0.65, -0.55, -0.53)$	0.00023	$(-0.64, 0.03, 0.77)$
weeds	0.011	$(0.60, 0.61, 0.52)$	0.00045	$(-0.04, -0.63, 0.78)$

Table 4.2: **The eigenvectors and eigenvalues of a single test image** *For each class, λ_0 is the dominant eigenvalue, and Λ_0 is the eigenvector corresponding to it. Similarly, λ_1 is the second-largest eigenvalue, and Λ_1 is its eigenvector. All three classes of data have a dominant eigenvector near the $r = g = b$ line, which implies there is little useful information along this dimension. A 2D classifier obtained by projecting the data down the $r = g = b$ line should be as effective as a classifier in 3D space. This correlation of pixel intensities is probably the reason the intensity-based classification performed so poorly. All intensity data has been normalised to lie in the range $[0, 1)$, and the smallest eigenvector/eigenvalue pair of each class have been omitted from the table.*

near the line $r = g = b$. This indicates that the measurements give little information in that direction, which presents an idea for a potential classification scheme for the data.

4.3.3 A linear classifier in RGB space

The dichromatic reflection model (DRM) predicts that the perceived colour of classes of differently coloured objects should lay on distinct plane segments in RGB space. Little information is contained in the $r = g = b$ direction. The covariance matrices of the data agreed with this prediction; they had a strong eigenvalue corresponding to an eigenvector near the $r = g = b$ line, and the differences between mean values of each of the classes in that direction were small compared to the eigenvalue. A minimum-distance classifier in three space for a system such as this would be equivalent to a minimum-distance classifier in 2-space where the third degree of freedom was removed by projecting orthographically down the dominant eigenvector (the one nearest $r = g = b$).

In such a system, the decision boundary of the minimum-distance classifier separating two classes of data in 3D will be a plane approximately perpendicular to the line joining the pair of class means. These three lines define a plane that also contains the three class means. The decision boundaries will all be perpendicular to this plane, since they are each perpendicular to a line contained in the plane. If the 3-space is orthographically projected down onto the plane containing the class means, the decision boundaries will be mapped

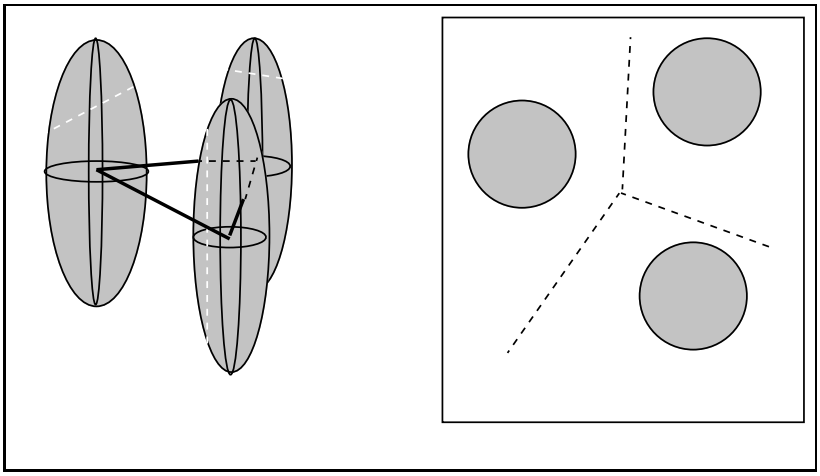


Figure 4.4: **2D Projection of a 3D minimum distance classifier** *Minimum distance classifiers in a three class discrimination problem have the property that there is an orthographic projection into 2D space that preserves the classification. This can be seen by looking at the three distributions in the left figure. The classification boundary between each pair of classes is a plane perpendicular to the line joining the mean of the two classes. Since the three means define a plane, and the lines joining each pair of means lies in this plane, the three classification boundary planes will be perpendicular to this plane. By choosing an orthographic projection into 2D that maps the three boundary planes onto lines, the essence of the classification scheme is preserved but in a discrimination scheme in 2-space.*

onto lines, and points in a classification region in 3-space map onto an analogous region in 2-space. Little information is lost by the reduction of dimension of the problem. (see figure 4.4).

Such a classifier was implemented and evaluated (see figure 4.5). For each image, the covariance matrix of the image and its dominant eigenvector was computed. Additionally, *a priori* information about the location of the means of each of the three classes was computed. Points were then classified by associating data point c with class ω_i if c was closer to μ_i than all the other classes.

On the average, 84% of the pixels in the test images were correctly identified, with a best case of 95% and a worst case of 68% (see table 4.3). A simple weighting scheme was also tested as an extension to the classifier to compensate for the difference in the class priors. The size of the region of the image covered by weeds was not the same as the size of the region covered by either soil or cauliflowers, and furthermore, the proportions of each

changed from image to image. For each image, a wide range of weights was tested and the best classification score over all weights was recorded. Classification improved to 89% on the average, with a best score of 97% and a worst of 82% (see table 4.3). As expected, the weights varied significantly between images.

4.3.4 Summary

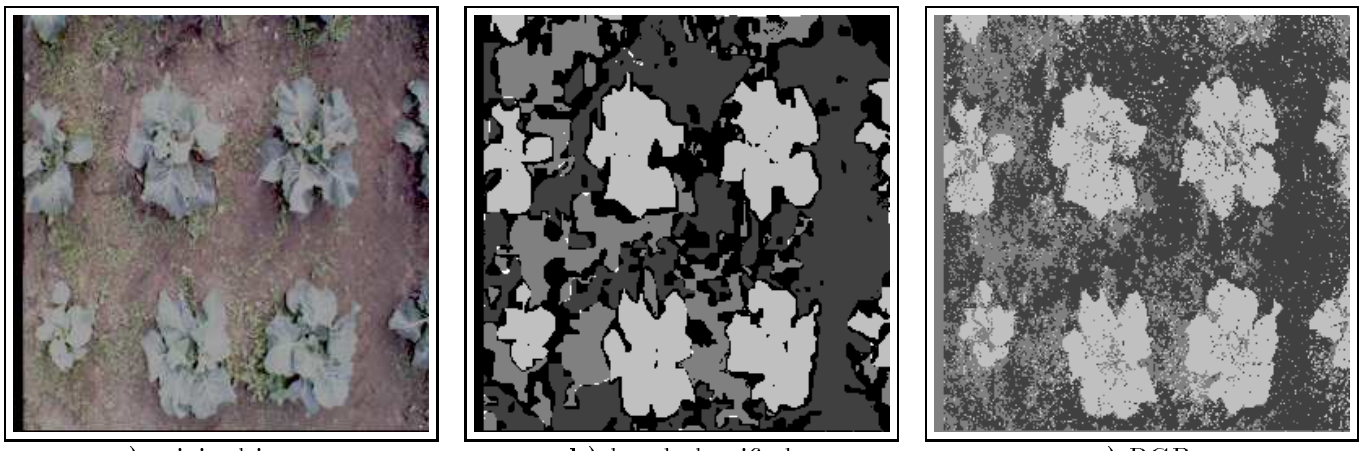
The problem of discriminating cauliflowers from weeds and soil using RGB measurements was studied and a simple linear classifier was implemented. Its performance was reasonable, but it required *a priori* knowledge of the location of the class means of the data. The location of the class means, as well as the class priors, varied significantly between images, which implies static training information may not be of much use. To implement such a scheme as an automatic classifier, a method would have to be developed to update estimates of the class means online.

One such method was used by Crisman (Crisman, 1992) for colour-based recognition of roads under various lighting and weather conditions. Higher-level reasoning functions identified regions of the image that were known with a high probability to belong to a particular class. The colours in these regions were measured and used to update the estimates of class means and priors.

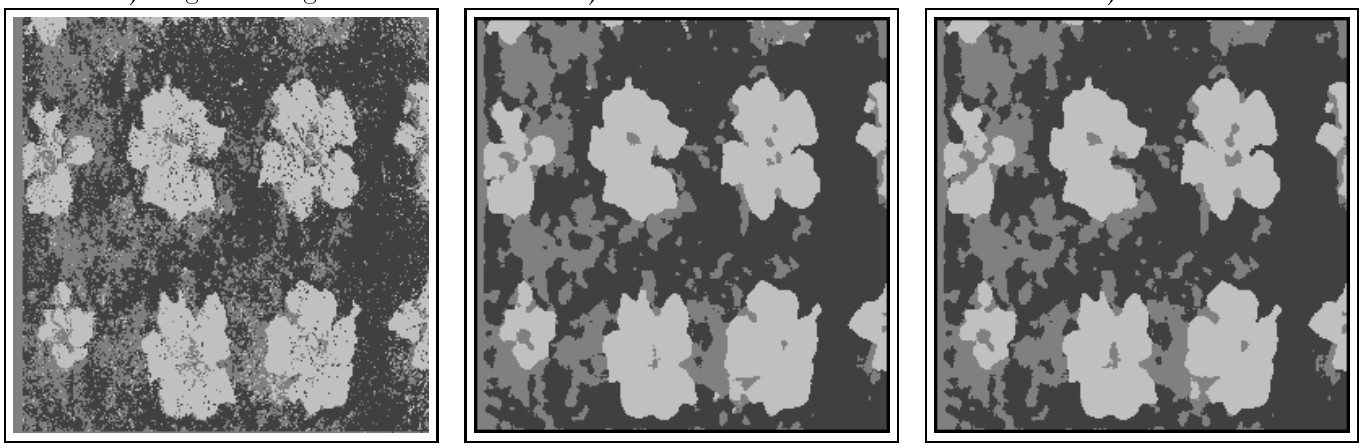
Since a system like Crisman's is difficult to implement, it is important to look at other potentially simpler schemes to compare the performance of the classification mechanism without taking into account adaptive systems designed to compensate for weak classifiers.

4.4 Classification of field cauliflowers in hue space

Colour has proved useful for classifying the data in our problem domain and a simple classifier based on RGB measurements has been constructed and tested. Since the measurements of the data seem to be significant in only two of the three dimensions, it is interesting to consider other representations of colour that allow a more natural reduction of the dimen-



a) original image | **b**) hand classified | **c**) RGB



d) weighted RGB | **e**) mode filtered | **f**) weighted and mode filtered

region type (image e)	classified as (%) cauliflower	soil	weed
cauliflower	96.85	0.35	2.79
soil	0.16	98.57	1.27
weed	2.79	17.67	79.54

Figure 4.5: **Example of linear RGB discrimination** *In all images except* **a**)*, black represents unclassified regions, dark grey represents soil, medium grey represents weeds, light grey represents cauliflowers, and white (* **b**) *only) represents regions misclassified as belonging to more than one class. The significant regions of black in image* **b**) *indicate that in parts of the image is was not possible to confidently classify the pixels by hand. Classification was done by a simple minimum distance classifier in RGB space. Adding a linear weighting to the means to reflect the relative class sizes improves classification in many images, though not in this particular one. The* 3×3 *mode filter* **e**) *and* **f**) *provides a significant improvement in classification performance. The confusion matrix for image* **e**) *is shown. As with hue-based classifiers (see figure 4.6) discriminating the weeds is the most difficult.*

sion of the measurements. In the HSV representation of colour, v (value) is proportional to the projection of the point on the $r = g = b$ line. By only considering s (saturation) and h (hue), the measurement space is naturally reduced to two dimensions.

The HSV representation also has the advantage h (hue) quantifies the human perception of hue. Since the different classes of plant-life are perceived to have distinct hues from the ground, the h measurement might be a good way to distinguish them. Classification based on a scalar measurement instead of a vector measurement is also desirable, because many low-level vision techniques work best on scalar measurements.

4.4.1 An efficient algorithm for finding minimal risk in hue space

As the first step in evaluating the usefulness of hue as the sole measurement for classifying the test images, a Bayes decision rule was computed. To find a more realistic upper bound on the accuracy, the decision rule was restricted so that each class (cauliflower, soil, or weed) lay in a single contiguous region of hue space. An efficient algorithm was developed to find these optimal regions.

The problem of determining the optimal regions for hue-based classifications has two parts. First, the boundaries between the three regions must be determined. Since hue-space is one-dimensional but cyclic, the boundaries between three regions are specified by a total of three points. By dividing hue-space into $(\frac{360^\circ}{n})$ intervals, there are n^3 combinations of boundaries to consider. Once the location of the boundaries is determined, the matching between the classes and the regions must be chosen.

Since the loss function in the basic classification problem is the total number of misclassified pixels, a partial error can be computed after fixing the first two of the three region boundaries. This completely defines one of the regions, which is arbitrarily assigned to be the region of hues specifying cauliflowers. Once the cauliflower region has been fixed, any pixels corresponding to soil or weeds within this region have been misclassified, as have cauliflower pixels lying outside of the region. If the partial error at this stage exceeds the

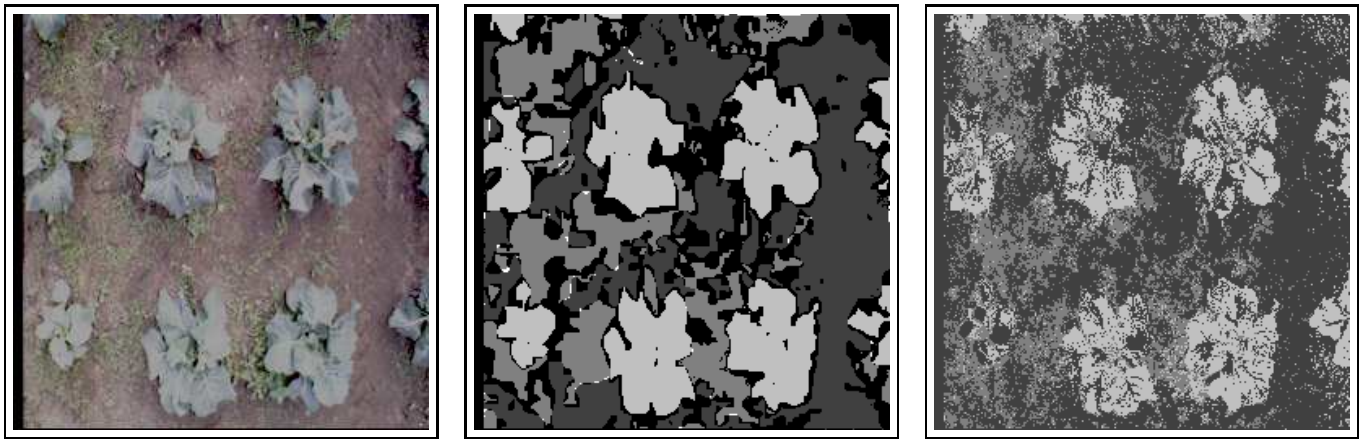
total error in the best complete classification so far, the boundary between the remaining two regions need not be considered. Only when the partial error is less than the best total error so far is the third boundary decided and the mapping between the remaining two classes and the remaining two regions chosen. By making a good first guess (the first guess for all images was $[120°, 270°)$ for cauliflowers, $[270°, 30°)$ for soil and $[30°, 120°)$ for weeds) the boundaries of the optimal regions were quickly (an average of five seconds on a Indy R4000PC) computed to within a degree.

Figure 4.6c shows the output of this classifier on image 2, where it correctly classified 76% of the image. The upper limit of the accuracy of the classifier on the test data was found to be between 76% and 94% depending on the image (see table 4.3).

4.4.2 Application of posterior white balancing to decrease risk of hue-based classification

For reasons outlined in section 3.3.3, the performance of a hue based classifier will be optimal when the colour of the light source in the image is on the line $r = g = b$. This occurs when a camera is white balanced. As a way to improve the results of hue-based classification, a white-balancing filter was run over the images, and the classification results were compared.

A video camera is normally white balanced by taking a photograph of a grey card, and then computing with corrective multiplicative constants so that the image pixels become a neutral grey (from Langford in (Langford, 1984)). Langford further states that a camera can be approximately white balanced without having a photograph of a grey card by assuming that the average pixel intensity across the entire image is a neutral grey. This method is used quite successfully in commercial film processing to remove the colour tinting associated with taking photographs under different lighting conditions (i.e. sunlight, fluorescent light or incandescent light). However, if the image being sampled is predominantly a single colour of high saturation, the correction will be incorrect and the image will acquire a new tint and look worse than before. For example, a small white cat on a large green lawn will become a magenta cat on grey-green grass (from (Langford, 1984)).



a) original image **b)** hand classified **c)** hue

d) white-balanced **e)** 3×3 mode filter **f)** white-balanced and mode filtered

region type (image f)	classified as (%) cauliflower	soil	weed
cauliflower	98.72	0.30	5.92
soil	0.32	98.80	20.15
weed	0.96	0.89	73.93

Figure 4.6: **Example of hue-based discrimination** *In all images except* **a)**, *black represents unclassified regions, dark grey represents soil, medium grey represents weeds, light grey represents cauliflowers, and white (* **b)** *only) represents regions misclassified as belonging to more than one class. The significant regions of black in image* **b)** *indicate that in parts of the image is was not possible to confidently classify the pixels by hand. Posterior white balancing* **d)** *and* **f)** *significantly improves the regions of the cauliflowers selected. The* 3×3 *mode filter* **e)** *and* **f)** *helps correct occasional misclassifications within large regions of correctly classified points. The confusion matrix for image* **f)** *is shown. Of the three classes, discriminating the weeds is the most difficult, partially because of its small prior, and partially because it is difficult to confidently identify regions of weeds for the training sets.*

Correct white-balancing is important for successful hue-based discrimination. To find a colour's hue, the colour is represented by a linear combination of white and a pure colour (Foley et al., 1990). White-balancing a camera is essentially defining what colour that "white" is. A different white can greatly affect the hue assigned – especially with less saturated colours (see figure 4.7).

Using posterior white-balancing, an improvement in hue-based classification was gained in half of the images see table 4.3. In the images where classification improved, it was by an average of 4.6% with a best case improvement of 10.7%. In the images where accuracy decreased, the number of correctly classified pixels dropped by an average of 1.4% with a worst case of 5.2%. These images (images 6,9,11 – see table 4.1) were all overexposed, and much of the colour information had been lost. Getting a reasonable exposure from the cameras is difficult because of the amount specular reflections in many of the images. This is a problem that has not been adequately resolved.

Posterior white balancing isn't as useful as a true white balancing with a grey card. It is not good to assume that the average colour of an image taken of a cauliflower field is grey. At certain times in the growth cycle the ground will dominate the image, giving an overall red tint to the image, and at other times, the plants will dominate the scene, biasing any white balancing towards green. By mounting a grey card in front of the camera to partially obscure the ground scene, a true measurement of the colour of ambient light could be taken, and the camera could be continually adjusted for changes in the colour of sunlight relating to the presence or absence of clouds. This could also be used for feedback to the iris to control the aperture of the camera. To date neither idea has been implemented.

4.5 Improving classification through local information

One problem with pixel-based classification is that it is ignoring a wealth of information available for classification. In the same way that it is difficult to understand what's behind a door by looking through the keyhole, the problem of classifying an image is much easier

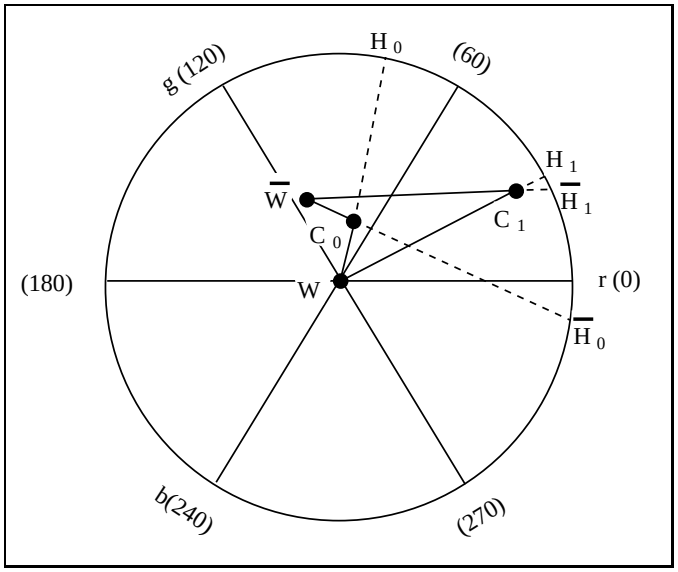


Figure 4.7: **Hue-based discrimination is sensitive to white balancing.** *Describing colour as the linear combination of "white" light and a pure colour, the hue of the colour is defined to be the "angle" of the pure colour. On a correctly white-balanced camera, the colour of a neutral (grey) card will lie exactly in the centre of the distribution (at W), and so the hue of colour C_0 will be correctly measured as being H_0, and the hue of colour C_1 will be correctly measured as being H_1. If the camera is not correctly white balanced, the colour of the white card will move away from the centre of the distribution (point $\bar{W}$). This means that, in the old frame of reference, the "white" point used in the linear combination has also moved away from centre, thus skewing the pure colour needed to solve the linear combination. The differences in the measured hue can be substantial. Since colour C_1 is fairly saturated, its measured hue is affected only slightly, moving from approximately* $25°$ *(H_1) to* $30°$ *($\bar{H}_1$). However colour C_0, which is not very saturated, is greatly affected by the change, with its measured hue changing from approximately* $80°$ *(H_0) to* $350°$ *($\bar{H}_0$).*

when one considers the image as a whole instead of one pixel at a time. It is the classic case of the aggregate being greater than the sum of its parts.

There are several standard ways of incorporating region-based information into classification schemes. Region growing techniques (Brice and Fennema, 1970; Zucker, 1976) group pixels of like characteristics together until the image is completely segmented. Usually "like characteristics" mean some form of ad hoc local differential measure: e.g. two neighboring pixels are considered to be in the same region if their intensities are within a certain threshold of each other.

The primary problem with region growing methods are that they aren't particularly fast. In a real-time tracking application, speed is at a premium. A second problem is that

region growing, like most ad hoc methods, requires quite a bit of fine tuning of parameters.

Markov Random Fields (MRFs) (Geman, 1987) are a second method for region-based classification. Instead of using heuristics like region growing techniques, MRFs pose the problem in Bayesian terms. Each pixel of the image is an observation of a random variable. The random variables are on a 2×2 grid, and the likelihood that the variable is in a given state is conditional on the states of the random variables in its neighborhood (usually 3×3 or 5×5). Through training sets, it is possible to learn the p.d.f.'srelating the variable to its neighborhood and the observation (the pixel) to the state of the variable (the class). Then the classification problem becomes finding the most likely set of states given an observation.

The largest drawback of MRFs is that they are very slow. In practice (Geman, 1987; Geman and Geman, 1984) MRFs are solved through iterative, stochastic methods, such as simulated annealing. These methods may require hundreds of passes and are not suitable for real-time systems.

As an alternative to these two global solutions, a simple smoothing method was implemented that post-processed the output of the single pixel classifier. Since the features (i.e. the plants) in the image are all at least 25 pixels in area, it is almost certainly an error when the classifier identifies a pixel as crop, yet classifies all of its neighbors as soil. A smoothing filter will correctly reclassify such a pixel as soil.

Normally, for this kind of smoothing problem, a median filter is used (Bangham, 1993). It eliminates isolated misclassifications without moving the boundaries of the classification regions. However, a median filter is only meaningful when the data are ordered. In the case of smoothing classification maps, the filter must find the 'average' classification for a region; the classes don't have an order (i.e. you can't say a weed is greater than a cauliflower or soil) so finding their median is meaningless. An alternative in this situation is a mode filter. A mode filter is just like a median filter except that the mode operation is applied instead of the median operation. It has many of the same desirable properties as a median filter, yet it doesn't have the restriction that the data are ordered.

A 3×3 mode filter was applied to the classified images. It replaced the classification of each pixel with the most common classification in the 3×3 neighborhood surrounding it. In the test images, the mode filter improved classification in almost all cases (see figures 4.5 and 4.6 and table 4.3). In some cases, the mode filter even outperformed the Bayes classifier. This is possible because the Bayes classifier makes a classification decision based on a single pixel: the mode filter incorporates local information, so its performance is not bounded by the Bayes classifier.

The speed of the mode filter is also appropriate for real-time applications. The mode of a neighborhood takes some additional time to compute, but this increment is small especially compared with the global classification methods mentioned earlier.

4.6 Analysis of Results

While theory may suggest which classifications methods may or may not be good for the data in the test set, the ultimate measure is relative performance. The two Bayes classifiers and the eight most practical classification schema were tested and compared. Table 4.3 shows their performance on each of the data sets. The ten methods are most easily broken up into three groups: the Bayes classifiers, the linear RGB classifiers, and the hue-based classifiers.

4.6.1 Description of the classifiers tested

Both of the Bayes classifiers were included in the test: the Bayes classifier for RGB images (section 4.3.1) and Bayes for monochrome images (section 4.2). They establish the upper performance limit for any classifier that makes a decision based on a single pixel of data.

The linear RGB classifiers projected the 3D colour information down the line $r = g = b$ and classified data based on their distance from each of the class means (as in section 4.3.3). The simple weighting scheme described at the end of the section was also tested. The performance of both methods was tested with and without the 3×3 mode filter described

	Bayes classifier		3D to 2D projection				Hue-based methods			
image	rgb	black & white	plain	mode	linear weight	lin. & mode	hue	hue & mode	white bal.	w.bal & mode
1	97	64	77	82	77	83	89	96	92	97
2	86	65	84	94	84	94	76	85	84	93
3	98	64	94	95	94	96	93	95	93	95
4	93	63	81	85	81	85	84	90	85	91
5	96	68	83	88	84	88	87	91	88	94
6	98	80	83	87	87	89	86	88	85	89
7	99	73	77	79	94	96	88	91	96	97
8	97	67	68	77	78	84	77	82	77	83
9	99	95	92	97	95	96	95	96	94	94
10	99	93	93	97	95	97	90	90	90	90
11	99	74	95	96	95	96	94	95	89	93

Table 4.3: **Comparative results of all methods on the eleven sample images** *All values are the percentage of pixels labelled correctly. While the percentage of pixels correctly labelled is not the best measure of how well the image segmentation algorithm worked (for example, the black and white version of image 9 was classified 95% correctly by the Bayes classifier, yet only 2.44% of the cauliflower pixels were labelled correctly) it does provide a good overall way to compare the different methods attempted for image segmentation.*

in section 4.5.

The hue-based classifier (section 4.4) and hue-based classifier with posterior white balancing (section 4.4.2) were also tested on the data. The performance of both was tested with and without the 3×3 mode filter.

4.6.2 Comparison of performance

The accuracy of all classification schema on all images is in table 4.3. The losing method is clear. The accuracy of the black and white Bayes classifier was much lower than any of the colour-based methods: an average of 11% lower than the simplest linear RGB classifier, and an average of 14% lower than the simplest hue-based classifier. Since the Bayes classifier is the upper limit for any single-pixel classifier, it means that there is insufficient data to classify the black and white images.

On the other end of the spectrum, the mode filter improved classification by an average of 4% across all images and classifiers. In some cases, classification accuracy after the mode filter exceeded that of the RGB Bayes classifier. This is possible since the mode filter

smooths the classification map, thus using extra region-based information unavailable to the Bayes classifier.

The remaining question is: to which single pixel classifier should the mode filter be applied? The RGB Bayes classifier can't be used in practice since it requires prior knowledge of the correct classification (thus negating the whole purpose for using a classifier). Of the remaining methods, the performance difference was smaller. The linear RGB classifier with mode filter averaged 89% classification, as compared with 91% for the hue-based classifier with mode filter, but on almost half the images former performed at least as well as the latter. The data were more contradictory for the weighted linear RGB classifier with mode filter (91% average accuracy) and the white-balanced hue-based classifier with mode filter (92% average accuracy): the latter averaged 1% better classification, yet the former was the best classifier in more images (six to four with one tie).

Since the accuracy of the linear RGB and hue-based methods is comparable, other factors must be considered when making the decision as to which is best. One advantage that the linear RGB methods have over their hue-based counterparts is stability. The singularity in the formula for hue means that pixels near the line $r = g = b$ can be mapped to widely varying hues depending on the image noise and white balancing of the camera. This is especially important in extreme lighting conditions: as the image becomes over or underexposed, colours become less saturated and thus hue-based classification will be more sensitive to noise. The 3D to 2D map used by the RGB classifiers is linear and thus has no singularities, so the RGB classifier should be more stable in extreme lighting conditions.

Table 4.3 bears this out. Images 9, and 11 are both over-exposed. Images 3 and 10 are very underexposed. In all four cases, the linear RGB classifier was better than the hue-based classifier. On image 1, the image with the best colour the hue-based classifier outperformed the linear RGB classifier by 14%. Since the goal is to classify images taken in uncontrolled lighting, poor exposure is the rule instead of the exception. For this reason, the weighted linear RGB classifier was chosen to be the best for the plant classification

problem. However, the accuracy of the method is still not consistently high enough to be used by itself as a classification technique. Chapter 5 explores methods to combine the temporal continuity and robust shape model of Dynamic Contours with the single-pixel weighted linear RGB classifier in order to obtain a more robust classification scheme.

4.7 Conclusion

The upper bounds of classification based on black and white, RGB and hue measurements were found and compared using a test data set. The data came from a database of video images of cauliflower fields, where the cauliflowers were at various states of growth and the lighting conditions varied. The accuracy of linear classification schemes based on a reduced dimensionality RGB space and hue space were compared. The two methods compared well to each other, but the weighted linear RGB classifier was judged superior because it was more robust in bad lighting conditions. None of the methods considered were accurate or reliable enough for use by themselves. Their accuracy was high enough that they could serve as a means to increase contrast for a subsequent vision scheme.

The accuracy of the classifiers was significantly raised by incorporating information about the spatial continuity of the image by using a mode filter. Dynamic contours (Curwen, 1993) with their more sophisticated spatial and temporal models could provide additional information to reduce the error and provide acceptably accurate labelling and tracking of the different classes in the test images.

5

Modelling shape and motion independently in an Active Contour tracker

The performance of Active Contours in tracking is highly dependent on the availability of an appropriate model of shape and motion, to use as a predictor. Such models can be hand-built, but it is far more effective and less time-consuming to learn them from a training set. Techniques to do this exist both for shape, and for shape and motion jointly. This chapter extends the range of shape and motion models by modelling jointly the random variations in shape arising within an object-class with those occurring during object motion. The resulting algorithm is applied to tracking of plants captured by a video camera mounted on an agricultural robot (see figure 1.1). In addition, previously known algorithms are adapted to correctly learn dynamics from short training sequences and multiple training sequences.

5.1 Introduction

The use of Kalman filters (Bar-Shalom and Fortmann, 1988) to track the motion of objects in real time is now a standard weapon in the arsenal of Computer Vision (Dickmanns

and Graefe, 1988; Stephens, 1990; Lowe, 1992; Gennery, 1992; Harris, 1992b; Terzopoulos and Szeliski, 1992; Blake et al., 1993b). A crucial consideration for effective real-time performance is that some form of dynamical model be identified (Pentland and Horowitz, 1991; Blake and Isard, 1994; Baumberg and Hogg, 1995b) and used as a predictor. In many cases, available models are deterministic — based on ordinary differential equations. However, to be usable in a Kalman filtering framework it is crucial that the model contain both deterministic and stochastic components; stochastic differential equations are used instead. Such stochastic models can be learned effectively from training data (Goodwin and Sin, 1984; Blake and Isard, 1994).

In this chapter we will develop two significant elaborations for stochastic dynamical models. The first concerns a correction to the learning equations that becomes important when short training sequences are used for learning. The second concerns modelling object classes for objects in motion.

5.2 Active Contours

The tracking problem in computer vision is that of fixating a point on a moving object (Aloimonos et al., 1987), i.e. determining the position and velocity of an object in a sequence of images. Active Contours go about solving the tracking problem from a model-based approach. The object's shape and expected motion are specified ahead of time. Additionally, Active Contours only attempt to solve the tracking problem in the image frame instead of the real-world frame: the contours will determine where the tracked object is in the image but make no attempt to determine the physical location of the object.

One of the fundamental problems in tracking is the sheer volume of data. A standard colour PAL television signal sampled with an aspect ratio of 1:1 contains 33MB/sec of data; NTSC contains 30MB/sec of data. Global computations on such a volume of data are prohibitively expensive. As an example, to apply a 3×3 convolution to a PAL signal in real time requires a machine capable of 190MFlops with a bus that can support data

transfers of at least 66MB/sec. Active Contours address this problem by using foveal vision, i.e. identifying small regions of interest in the image before analysing the image, and only processing those regions. Through this technique, the number of pixels analysed can be greatly reduced (the examples in this chapter required analysing approximately 45,000 pixels per second).

To keep the regions of interest as small as possible, Active Contours only track the outlines of the objects. The Active Contour tracker uses N span B-spline image curves to follow tracking targets in images. While the method is not restricted to use a particular order or kind of spline, only quadratic (order 3) B-splines were used for results in this thesis. B-splines have certain mathematical advantages (such as locality) that make them more efficient to use with the Active Contours' Kalman filter. Quadratic B-splines were chosen over those of other orders because linear splines (order 2) required too many control points to define a given contour and cubic (order 4) splines were significantly more complex, yet added little expressive power.

A given B-spline curve is represented as

$$\mathbf{r}(s,t) = (x(s,t), y(s,t)) = (B(s)\mathcal{X}_t, B(s)\mathcal{Y}_t) \tag{5.1}$$

where s is the parameter that parametrises the spline at a specific time t, $B(s)$ is the standard B-spline matrix (Bartels et al., 1987), and $\mathcal{X}_t$ and $\mathcal{Y}_t$ are the vectors of the x and y coordinates of the control points at time t. The Active Contour tracker generates estimates of these control points $\hat{\mathcal{X}}_t$ and $\hat{\mathcal{Y}}_t$ with the goal that the B-spline parametrised by them closely matches the target in the image. The circumflex (hat) accent on $\hat{\mathcal{X}}$ and $\hat{\mathcal{Y}}$ indicate that they are only estimates of the (unknown) true values $\mathcal{X}$ and $\mathcal{Y}$.

Figure 5.1 shows a simple quadratic B-spline with the control points $\mathcal{X} = \begin{pmatrix} -2 & -2 & 2 & 2 & 0 \end{pmatrix}^T$ and $\mathcal{Y} = \begin{pmatrix} -1 & 1 & 1 & -1 & 1 \end{pmatrix}^T$. The overscore is used with $\overline{\mathcal{X}}$ and $\overline{\mathcal{Y}}$ to indicate that the template is some sort of a mean or average shape. This B-spline is also the template (base shape) for figure 5.2.

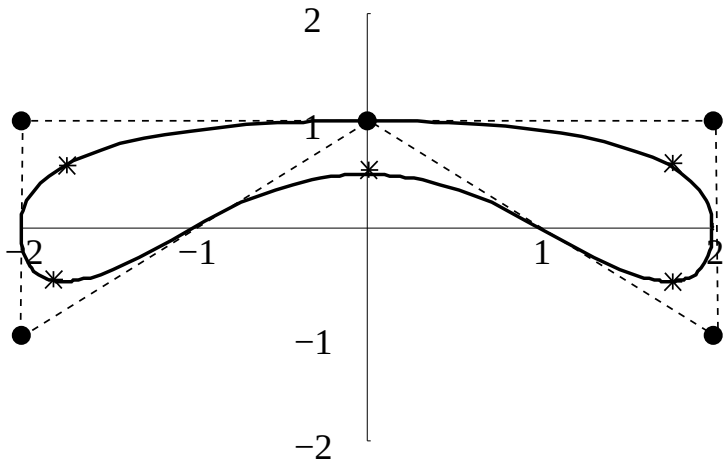


Figure 5.1: **A B-spline contour with enclosing polygon and control points** *This figure shows a spline (dark line) defined by the template* $\mathcal{X} = \begin{pmatrix} -2 & -2 & 2 & 2 & 0 \end{pmatrix}^T$, $\mathcal{Y} = \begin{pmatrix} -1 & 1 & 1 & -1 & 1 \end{pmatrix}^T$. *The five control points are shown by dots, and are connected by the dotted polygon to show their order. Since this is a quadratic B-spline, notice that the control points in general are not on the spline itself. Since it is often useful to have points of reference on the spline itself, the points on the curve most influenced by each of the five points are shown by crosses. These crosses can be used to visualise the parameterisation of the curve and to detect lateral motion of the B-spline (as in figure 5.6).*

One of the major advantages of the Active Contour paradigm over Snakes (as described by Kass, *et. al.* 1987) is the use of a shape space. Blake, Curwen and Zisserman (1993) showed that allowing the control points to vary independently of one another (i.e. $2N$ degrees of freedom) could cause the tracker to be unstable. Instead, they restricted the control points to deform within a linear space that had a lower dimension (6 to 8). Additionally, motion was defined to be relative to a template $\begin{pmatrix} \overline{\mathcal{X}}^T & \overline{\mathcal{Y}}^T \end{pmatrix}^T$. A point $\mathbf{X}$ in the 'state space' or 'W-space' was related to a point $\begin{pmatrix} \mathcal{X}^T & \mathcal{Y}^T \end{pmatrix}^T$ in the 'control point space' by the equation

$$\begin{pmatrix} \mathcal{X} \\ \mathcal{Y} \end{pmatrix} = W\mathbf{X} + \begin{pmatrix} \overline{\mathcal{X}} \\ \overline{\mathcal{Y}} \end{pmatrix} \tag{5.2}$$

where W is the matrix of basis vectors defining the degrees of freedom. These degrees of freedom were chosen so that B-splines in this subspace corresponded well with the expected appearance of the tracking target.

Similarly, points in control points space are related to points in the state space by the

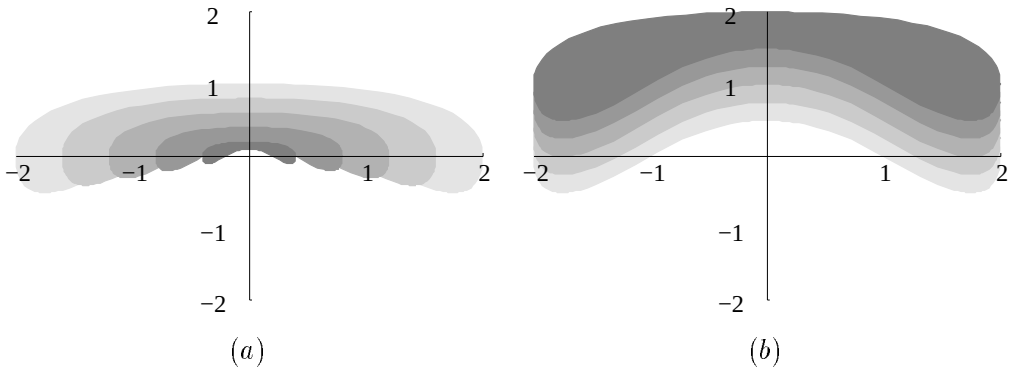


Figure 5.2: **The effects of changing X** *The system described in this figure has a template* $\overline{\mathcal{X}} = \begin{pmatrix} -2 & -2 & 2 & 2 & 0 \end{pmatrix}^T$ *and* $\overline{\mathcal{Y}} = \begin{pmatrix} -1 & 1 & 1 & -1 & 1 \end{pmatrix}^T$ *and a shape space of* $W = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ -2 & -2 & 2 & 2 & 0 & -1 & 1 & 1 & -1 & 1 \end{pmatrix}^T$. *Figure 5.2a shows how the shape of the spline changes as* $\mathbf{X}$ *varies between* $\mathbf{X} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}^T$ *(light grey) and* $\mathbf{X} = \begin{pmatrix} 0 \\ -0.8 \end{pmatrix}^T$ *(dark grey). Figure 5.2b shows how the shape of the spline changes as* $\mathbf{X}$ *varies between* $\mathbf{X} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}^T$ *(light grey) and* $\mathbf{X} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T$ *(dark grey).*

matrix M where

$$M \triangleq (W^T \mathcal{H} W)^{-1} W^T \mathcal{H} \tag{5.3}$$

$$\mathcal{H} \triangleq \int_0^N \begin{pmatrix} (B(s))^T B(s) & 0 \\ 0 & (B(s))^T B(s) \end{pmatrix} \tag{5.4}$$

Because the B-spline does not have a unit-length parametrisation, it is necessary to use the metric matrix $\mathcal{H}$ to convert distances in spline space to distances in the image. A more detailed explanation of M and $\mathcal{H}$ can be found in Curwen (1993).

Figure 5.2 shows a simple shape space based on the template defined in figure 5.1. By varying the values in $\mathbf{X}$, the spline scales (figure 5.2a), and translates in the y direction (figure 5.2b).

While W, $\overline{\mathcal{X}}$ and $\overline{\mathcal{Y}}$ completely characterise the possible configurations for the spline, Active Contours go one step further. They employ a predictive model of motion that characterises the likely location of the object based on its location in past frames.

Many researchers have found that linear stochastic second order difference equations are useful for representing the observed motion of an object in a visual tracking system (Blake

et al., 1995; Blake et al., 1994; Dalton et al., 1995; Baumberg and Hogg, 1995b). The joint dynamical model of the Active Contour is a second order stochastic difference equation:

$$\mathbf{X}_{n+2} - \overline{\mathbf{X}} = A_1 \left(\mathbf{X}_{n+1} - \overline{\mathbf{X}}\right) + A_0 \left(\mathbf{X}_n - \overline{\mathbf{X}}\right) + B\mathbf{w}_n. \tag{5.5}$$

This system is essentially equivalent to a damped pendulum driven by noise. The state (the position and shape) of the modelled outline contour at time n is $\mathbf{X}_n$, an N-dimensional state vector. $\mathbf{X}_n$ is in the W-space. To find the spline corresponding to $\mathbf{X}_n$, premultiply it by W. The state of $\mathbf{X}_n$ will vary about a mean configuration $\overline{\mathbf{X}}$ (the center of the pendulum's oscillation), which is a constant in this simple model. Coefficients A_0 and A_1 fix the deterministic component of the model, the natural frequencies and damping constants of oscillatory modes coupled across the parameter space (Pentland and Horowitz, 1991). The stochastic component of the model is a noise source $\mathbf{w}_n$, a vector of N independent unit normal variables, coupled into the system via the $N \times N$ matrix B.

Figure 5.2 shows the effects of the $\overline{\mathbf{X}}$ and B terms on a 1D second order difference equation. A_0 and A_1 are fixed to produce an undamped system with a cycle of approximately 20. Changing $\overline{\mathbf{X}}$ moves the centre of oscillation. Increasing B makes the system less and less deterministic. B quantifies how well the stochastic model is expected to model the motion of the object. When B is relatively small, the model is thought to be good and the predictions should be quite accurate. As B increases, the system can depend less and less on the model; the tracker must analyse larger search regions which makes it more susceptible to distraction and other tracking failures. However, if B is too small (the accuracy of the model is overestimated), the tracker will also fail; it will look in such a small region that it won't be able to find the object.

By grouping all of the $\overline{\mathbf{X}}$ terms, equation 5.5 can be simplified:

$$\mathbf{X}_{n+2} = A_1\mathbf{X}_{n+1} + A_0\mathbf{X}_n + (I - A_0 - A_1)\overline{\mathbf{X}} + B\mathbf{w}_n. \tag{5.6}$$

This makes $(I - A_0 - A_1)\overline{\mathbf{X}}$ a constant vector, reducing the computational cost at runtime.

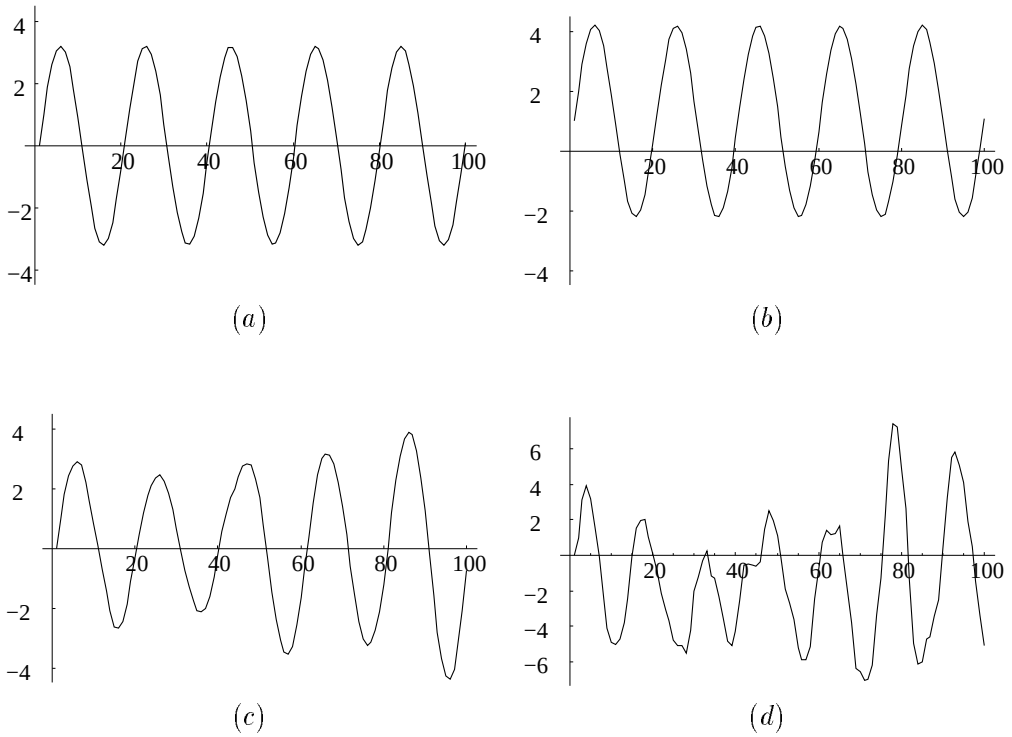


Figure 5.3: **The effect of B and $\overline{\mathbf{X}}$ on the dynamics** *The four figures above show the behaviour of a 1D system with $A_0 = -1$ and $A_1 = 1.9$, which in the absence of noise gives an undamped system with with a period of approximately 20. For all four figures, $\mathbf{X}_0 = \overline{\mathbf{X}}$ and $\mathbf{X}_1 = \overline{\mathbf{X}} + 1$. Figure 5.2a shows the basic system with no added noise ($B = 0$) and $\overline{\mathbf{X}} = 0$. In figure 5.2b, $\overline{\mathbf{X}} = 1$ so the centre of oscillation for the system has moved to 1. Figures 5.2c and 5.2d, show the effects of adding noise of $B = 0.1$ and $B = 1$ respectively. Notice that the period of the system is still approximately the same, but the short-term behaviour is less predictable. In an Active Contour, greater noise corresponds to a poor model, resulting larger search scales and degraded performance.*

Equation 5.6 can be rewritten in block-matrix form as

$$\begin{pmatrix} \mathbf{X}_{n+1} \\ \mathbf{X}_{n+2} \end{pmatrix} = \begin{pmatrix} 0 & I \\ A_0 & A_1 \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{X}_{n+1} \end{pmatrix} + \begin{pmatrix} 0 \\ I - A_0 - A_1 \end{pmatrix} \overline{\mathbf{X}} + \begin{pmatrix} \mathbf{0} \\ B \end{pmatrix} \mathbf{w}_n \tag{5.7}$$

which changes the second order SDE to a first order SDE of twice the dimension. The block-matrix form is the form the Kalman filter uses internally (see section A.1 in the appendix for more detail). In this case, the dynamics matrix is often written as

$$F \triangleq \begin{pmatrix} 0 & I \\ A_0 & A_1 \end{pmatrix} \tag{5.8}$$

making the equation

$$\begin{pmatrix} \mathbf{X}_{n+1} \\ \mathbf{X}_{n+2} \end{pmatrix} = F \begin{pmatrix} \mathbf{X}_{n} \\ \mathbf{X}_{n+1} \end{pmatrix} + \begin{pmatrix} 0 \\ I - A_0 - A_1 \end{pmatrix} \overline{\mathbf{X}} + \begin{pmatrix} \mathbf{0} \\ B \end{pmatrix} \mathbf{w}_n \tag{5.9}$$

5.3 Learning general shape spaces

The first process in setting up an Active Contour tracker is to decide what shape space to use. For many early results, the shape space chosen was the affine space (Blake et al., 1993b). This space is useful because its degrees of freedom describe the shape variability of a planar object being imaged by an affine camera (Mundy and Zisserman, 1992). The affine camera is a good model of a real camera when the depth of the objects in the scene is small when compared to the distance from the objects to the camera.

The six degrees of freedom in the affine shape space are independent x and y translation, independent x and y scaling, and two shearing motions. The effects of these six degrees of freedom on a template of a hand are shown in figure 5.4. Another advantage of the affine shape space is that it can be completely constructed from the template. The W matrix defining the affine shape space for a template $\begin{pmatrix} \overline{x}^T & \overline{y}^T \end{pmatrix}^T$ is

$$W = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \overline{x} & \mathbf{0} & \mathbf{0} & \overline{y} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \overline{y} & \overline{x} & \mathbf{0} \end{pmatrix} \tag{5.10}$$

where $\mathbf{1}$ and $\mathbf{0}$ are the vectors of all ones and zeroes respectively. The first two columns of the W matrix correspond to translation in the x and y direction, the third and fourth to scaling in the x and y direction, and the fifth and sixth vectors to the two shears. Figure 5.4 illustrates the effects of these degrees of freedom on a template of a hand.

5.3.1 Building general shape spaces through keyframes

Many times the affine shape space is not appropriate for tracking. In this case it may be necessary to construct a more general shape space from scratch. Constructing a general shape space requires slightly more work than constructing the affine shape space. Each column vector $\mathbf{Q}_i$ of the matrix W must be specified individually. There are several ways

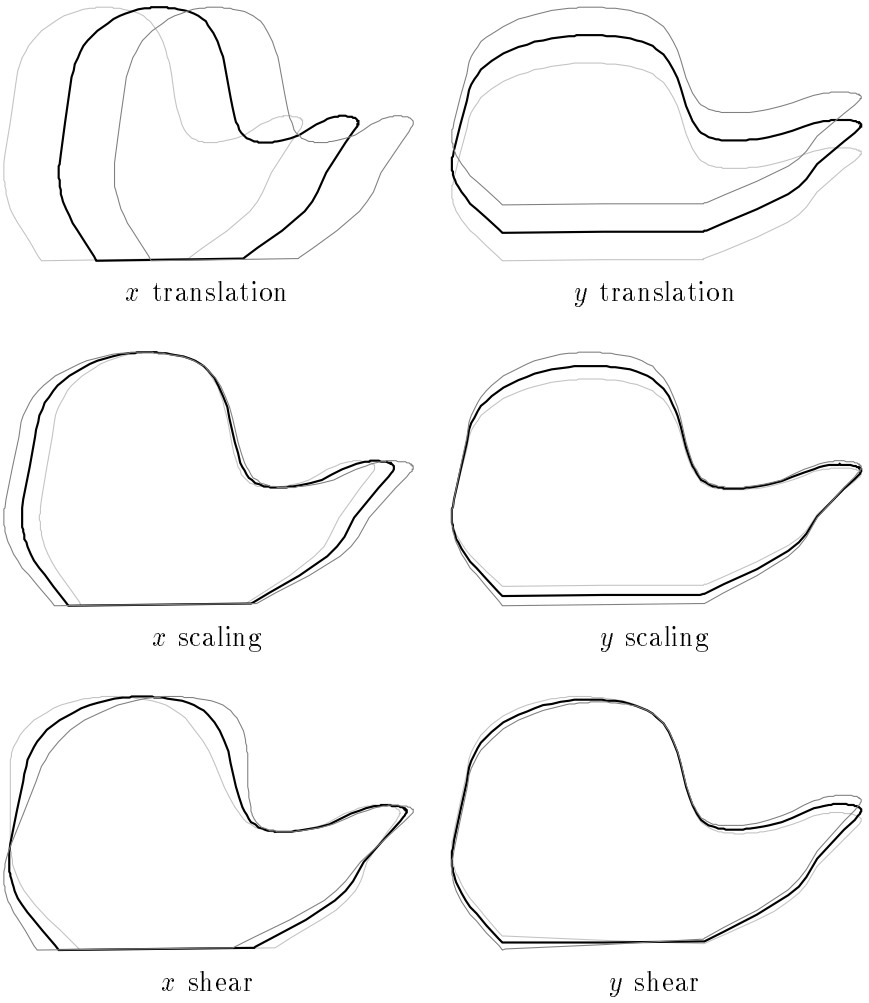


Figure 5.4: **An illustration of the six degrees of freedom of an affine shape space.** *The effects of the first four affine degrees of freedom (x and y translation and scaling) on the hand template are easy to visualise. The last two degrees of freedom (the shears) are easiest to understand by examining how they effect individual points, such as the corner in the bottom left of the template. The x shear fixes the y value of individual points whilst shifting their x value. The y shear does the opposite; it fixes the x value whilst changing the y.*

that this can be done. The first way is to fit B-splines to the target in individual frames in a training sequence. The shape space can then be defined to be linear combinations of these 'keyframes' (see figure 5.5).

One advantage of shape spaces built from keyframes is that the shape-space coordinates have clear interpretations. Figure 5.5, shows an very simple example of a keyframe shape

space. The template of the system is an outline of a left hand with the angle between the thumb and forefinger at approximately 45°. An outline of a left hand with an angle of about 30° between the thumb and index finger is used as a keyframe. Linear combinations of the template and the keyframe cause deformations in the B-spline like articulation of the thumb. To create a state-space that only allows this motion, W is set to have one column (one degree of freedom), where that column is equal to the difference between the control points of the keyframe and the control points of the template. Then by the formula

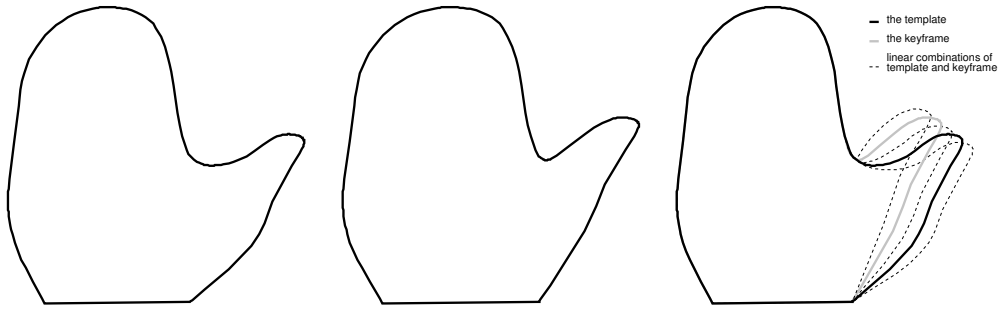
$$\begin{pmatrix} \mathcal{X}_t \\ \mathcal{Y}_t \end{pmatrix} = WX + \begin{pmatrix} \overline{\mathcal{X}} \\ \overline{\mathcal{Y}} \end{pmatrix} \tag{5.11}$$

Where X is scalar since it's a one-dimensional system. X has a simple interpretation: when $X = 0$, the B-spline looks like the template (about 45° between the thumb and forefinger). When $X = 1$, the B-spline looks like the keyframe (about 30° between the thumb and forefinger). In general, as X increases, the thumb approaches the forefinger, and as X decreases, the thumb moves away from it (figure 5.5c).

More ambitiously, the same two keyframes can be used to build a more versatile shape-space that allows for translation, zooming and rotation of the hand as well as the independent thumb movement. Minimally, this should require one parameter for the angle between the thumb and the hand, plus four parameters for the similarity transform, for a total of five parameters. However, the shape-space must be linear, so two parameters are required instead of one to express the angle between the thumb and the hand, and the shape-space is six dimensional. The template of the system is $\mathbf{Q}_1$ (the hand in a relaxed position with the thumb at a medium angle) and the shape matrix is

$$W = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{Q}_1^x & -\mathbf{Q}_1^y & \mathbf{Q'}_2^x & -\mathbf{Q'}_2^y \\ \mathbf{0} & \mathbf{1} & \mathbf{Q}_1^y & \mathbf{Q}_1^x & \mathbf{Q'}_2^y & \mathbf{Q'}_2^x \end{pmatrix} \tag{5.12}$$

where $\mathbf{Q}_1^x$ signifies the column vector of the x coordinates of the control points of the first keyframe, and $\mathbf{Q'}_2^x$ is the difference between the two keyframes of the x coordinates of their control points. Again, thumb articulation is naturally expressed by the shape-vector. By adjusting θ in the equation $\mathbf{X} = (0, 0, \cos\theta - 1, \sin\theta, b(\cos\theta - 1), b(\sin\theta))$ it is possible to



(a) original template (b) key frame (c) linear combinations

Figure 5.5: **An example of defining a shape space with keyframes** *Figure 5.5a shows a template of a left hand with the thumb extended. It is desired to construct a shape space that allows no motion of the hand except articulation of the thumb. To make such a shape-space, a B-spline is fit to a picture of a hand with a a slightly different thumb angle (figure 5.5). This second B-spline is known as a keyframe. Linear combinations of the template and the keyframe allow the B-spline to deform as if the thumb were articulating (figure 5.5c). A shape space with only this motion can be made by setting W to be equal to the difference between the template and keyframe. Then as the value of X changes in equation 5.2, the shape of the B-spline will change as in figure 5.5a to allow articulated thumb motion.*

rotate the hand; by adjusting b, it is possible to articulate the thumb in the same way as when the shape space was only the two keyframes.

5.3.2 Refining general shape spaces through PCA

When building a general shape space through aggregating keyframes, it is difficult to choose how many keyframes should be used to construct the shape space. Using too few keyframes results in a tracker that may fail because it is overly restrictive. Using too many results in a tracker that is unstable because it has too many degrees of freedom (Blake et al., 1993b).

A common solution (Kaucic et al., 1996; Cootes and Taylor, 1992) to this problem is to construct a shape space that is larger than necessary and then use principal components analysis (PCA). The classical approach to PCA would allow the following version of the approximation problem to be solved. Given a W matrix with N_W columns and a long training sequence with N_X data points such that $N_X > N_W$, solve:

$$\min_{W', \mathbf{Q}'_0, \mathbf{X}'_1, \ldots, \mathbf{X}'_{N'_X}} \left(\sum_{k=1}^{N_W} \left| \mathbf{Q}_k - \mathbf{Q}'_k \right|^2 \right) \tag{5.13}$$

where

$$\mathbf{Q}'_k = W'\mathbf{X}'_k + \mathbf{Q}'_0 \tag{5.14}$$

and $\mathbf{Q}_k$ is the original data point. The distance measure $|\ldots|$ is the Euclidean norm, that is, $|\mathbf{Q}|^2 \triangleq Q^T Q$. The well known solution to this problem gives $\mathbf{Q}'_0 = \overline{\mathbf{Q}}$, the mean of the training sequence, and W' is a matrix whose columns are the first N'_X of the orthonormal eigenvectors of the covariance matrix

$$S = \frac{1}{N_W}\sum_{k=1}^{N_W}(\mathbf{Q}_k - \overline{\mathbf{Q}})(\mathbf{Q}_k - \overline{\mathbf{Q}})^T, \tag{5.15}$$

ordering the eigenvectors in descending order of their eigenvalues. The eigenvalues are necessarily non-negative because S is a covariance matrix. The intuitive interpretation is that the eigenvalues represent variance in the training-set in the mutually orthogonal directions of the eigenvectors; the first N'_W eigenvectors form a basis for the subspace of dimension N'_X that 'explains' as much as possible of the variance in the training-set. Note that since the data $\mathbf{Q}_k$ all live within a shape-space of dimension N_X there will be at most N_X non-zero eigenvalues of S.

One problem with traditional PCA is that it maximises the fit of one set of the control points of the data to the control points of the vectors in the W matrix. However, in the same way that it was necessary to use the metric matrix $\mathcal{H}$ when converting from control point space to state space, it is necessary to use the metric matrix when computing a PCA. Instead of using the Euclidean norm, it is necessary to compute the principal components relative to the L_2-norm. In the same way that the Euclidean norm is defined by the relationship $|\mathbf{Q}|^2 = \mathbf{Q}^T\mathbf{Q}$, the L_2-norm is defined by $\|\mathbf{Q}\|^2 = \mathbf{Q}^T\mathcal{H}\mathbf{Q}$. The approximation problem for PCA using the L_2-norm should then be posed as follows:

$$\min_{W',\mathbf{Q}'_0,\mathbf{X}'_1,\ldots,\mathbf{X}'_{N'_X}} \left(\sum_{k=1}^{N_W} \left\|\mathbf{Q}_k - \mathbf{Q}'_k\right\|^2\right) \tag{5.16}$$

in which the L_2 norm $\|\ldots\|$ has been substituted where the Euclidean norm appeared before. The solution to this problem, which can be derived from the solution to the classical problem,

is that $\mathbf{Q}'_0 = \overline{\mathbf{Q}}$ as before and W' is a matrix whose columns are the first N'_X eigenvectors of the matrix $S\mathcal{U}$ where $\mathcal{U}$ is the symmetric square root of $\mathcal{H}$.

5.3.3 Results of learning shape spaces on real data

The PCA algorithm was tested on the difficult problem of characterising the shape of cauliflower plants. In this case, the B-spline template was chosen to have ten control points (a total of twenty degrees of freedom). Keyframes from thirty-six different plants were used to construct the general state space. Clearly the general state space was rank-deficient since it had more degrees of freedom than the underlying B-spline. The first six non-translational degrees of freedom in the PCA are shown in figure 5.6.

When used for tracking with an Active Contour, the eight dimensional PCA space was found to have similar tracking performance[1] to the standard affine shape space. Figure 5.7 shows snapshots comparing the performance of the PCA tracker and the affine tracker on real data. In the images shown, as well as in the rest of the sequences, there was no significant difference between the two. This is a surprising result, since the PCA space should have modelled the data better than the affine space. Many other authors have found PCA shape spaces to be very useful for tracking problems (Cootes and Taylor, 1992; Baumberg and Hogg, 1995a; Kaucic et al., 1996). Several different attempts were made to track with a PCA space, but tracking performance was never much better than the affine basis. As a result, the affine basis was used for the rest of the results dealing with plant tracking.

One possible alternative to using an explicit PCA basis is to allow the tracker to estimate the shape outside the W-space as well as inside the W-space. Normally an Active Contour tracker only allows the shape being tracked to assume shapes contained in the basis defined by the W-space. Sometimes it is desirable to allow the tracker to follow shapes that are 'close' to those in the W-space. This can be accomplished by running a second filter with

[1]For this discussion, performance is defined to be how well the B-spline matches in target in the image. In the absence of ground-truth information about where the object was, performance was judged by eye.

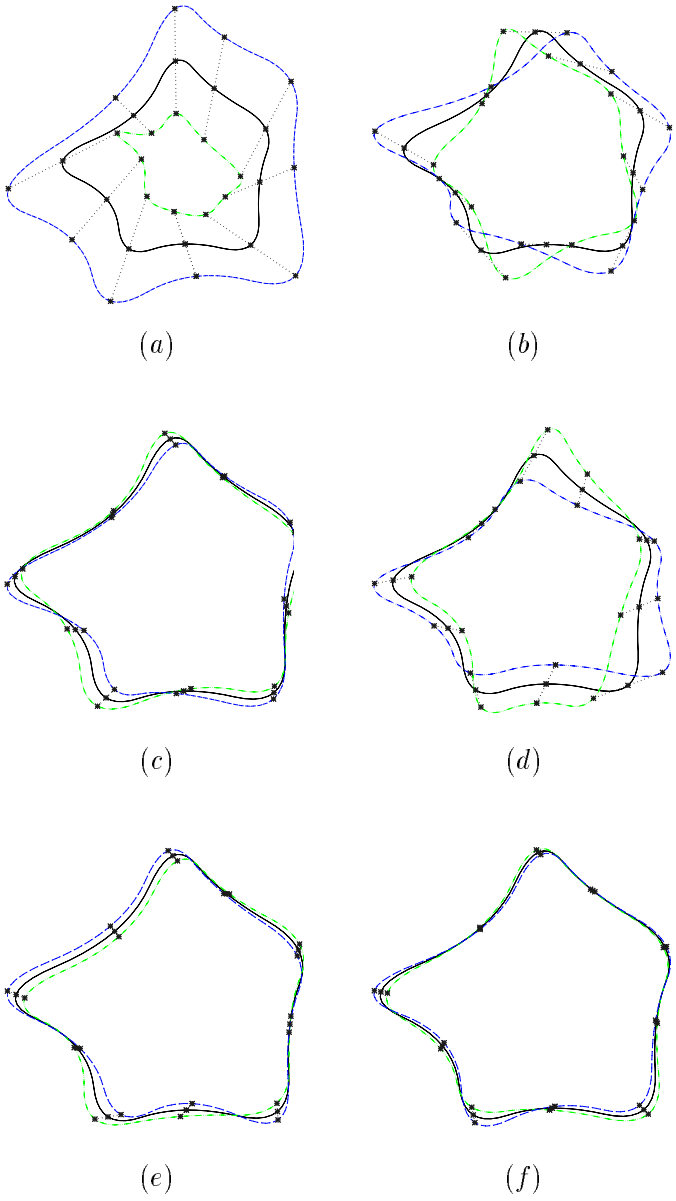


Figure 5.6: **PCA analysis of plants** *The principal components of the shapes of 36 different cauliflower plants were computed, and the six most significant modes of variation are shown here. In all six cases, the template is drawn as a solid black line, and the dashed lines show the template plus and minus two standard deviations of the particular mode. To show how the templates rotate during these transformations, crosses show the spots on the spline closest to each of the ten control points. The first several modes of variation are roughly affine deformations. The first mode clearly corresponds to global scaling, the second mode corresponds to a rotation, and the third mode is a horizontal scaling. The rest of the modes are non-affine.*

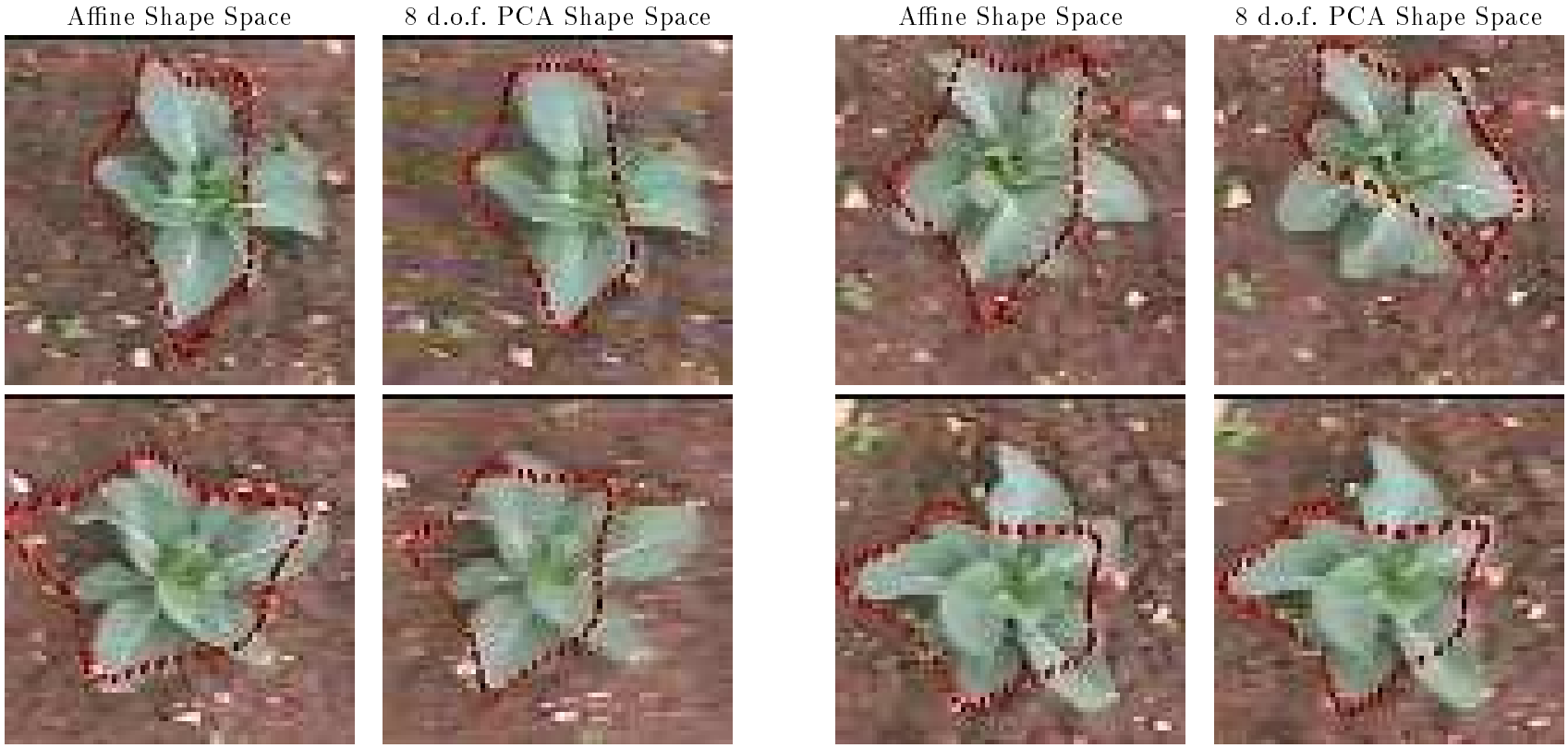


Figure 5.7: **A series of snapshots comparing shape spaces for tracking performance** *The affine shape space is compared with a PCA shape space for the difficult problem of tracking plants from a downward-pointing camera mounted on a tractor. Screenshots of the Active Contour using the affine space are shown along side of screenshots of the Active Contour using the 8 d.o.f. PCA space from figure 5.6. In general, there is little difference between the performance of the Active Contour using affine shape space and the Active Contour using the PCA shape space.*

the tracker, typically an alpha-beta tracker with very high measurement noise to keep the deviations from the W-space small. While extremely inelegant, this double-filter can occasionally be useful, and is a technique which should be investigated in a more rigourous manner.

5.4 Learning dynamics over short data sequences

The first problem addressed by this chapter concerns learning dynamical models from training sequences that are short when compared with the vibrational modes occurring in them. Second order linear differential and difference equations such as those described in (Blake et al., 1995; Blake et al., 1994; Dalton et al., 1995; Baumberg and Hogg, 1995b) can be characterised by the frequency and damping of their modes of vibration as well as their additive noise. In the past, most training sequences for learning have been long enough to observe multiple cycles of all vibrational modes (Blake et al., 1995; Blake et al., 1994; Dalton et al., 1995; Baumberg and Hogg, 1995b). When short training sequences are used for learning with the above methods, the modes of vibration can be learned incorrectly. The method described in this section adapts the learning algorithms to correctly find the vibrational modes even when the training sequences are short when compared to the frequency of the modes.

As stated before, Active Contours use second order SDEs to model and predict target motion. These SDEs are characterised by A_0 and A_1 (which set the frequency and damping of the vibrational modes), by B (which defines the uncertainty of the model), and $\overline{\mathbf{X}}$ (which determines the center of vibration). Several authors (Blake et al., 1995; Baumberg and Hogg, 1995b) have shown that by observing training sequences of data, it is possible to determine A_0, A_1 and B and thus "learn" the model that best describes the observed sequence. One point ignored by such methods is that of estimating $\overline{\mathbf{X}}$.

5.4.1 The mean of the sequence is not the mean of the system

Given a training sequence $\mathbf{X}_1, \mathbf{X}_2, \ldots \mathbf{X}_N$, the mean of the sequence is defined to be

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i \tag{5.17}$$

If this sequence has been generated by equation 5.6, there is a second 'mean' that is significant: $\mathsf{E}(\mathbf{X})$ where $\mathbf{X}$ is now considered to be a random variable and $\mathbf{X}_i$ are assumed to be samples from it. This mean is the mean of the system which generated the training sequence. In the case of a stable system, $\mathsf{E}(\mathbf{X}) = \overline{\mathbf{X}}$, while in the case of an unstable system, the mean is undefined.

Blake, *et. al.* and Baumberg and Hogg make the assumption that the mean of the training sequence is the mean of the underlying system that generated the training sequence. Many times this is a good estimate. When the system that generated the training data is stable,

$$\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i = \overline{\mathbf{X}}. \tag{5.18}$$

In other words, when the training sequence is long and the system is stable, the mean of the training sequence is approximately equal to the mean of the underlying system. Empirically, equation 5.18 is also a good approximation when the number of observed points (N) is large compared to the period system's modes or when the system is highly damped.

When the training sequence is both short and weakly damped, there is a systematic error in $\overline{\mathbf{X}}$ that can cause the rest of the terms to be misestimated. The errors cause large differences in the numerical values of the parameters and can change the overall characteristics of the system, causing previously undamped systems to be damped, and phase and frequency shifts to be introduced to the system.

Figure 5.8 shows the effect of such errors. The difference equation $X_{n+2} = 1.9998X_{n+1} - X_n$, an undamped sine-wave about 0 with a period of 446, is observed for several different periods of time that is small compared to its period. When the naive learning algorithm

is applied to the various observations, the learnt systems fundamentally differ from the original system. In figure 5.8a, a quarter cycle is observed. The system learned from this observation is highly damped, centered around 0.5, and its phase has been shifted from the original. In figure 5.8b, a half cycle is observed. The learned system has a similar period, yet it oscillates around a different mean. In figure 5.8c, three-quarters of a cycle is observed. This time the learned system is unstable and the oscillations increase with every cycle. As the training sequence becomes longer, the learnt dynamics more closely approximate the original dynamics (figure 5.8d and 5.8e).

5.4.2 Learning algorithm for single short sequences

This section adapts work by Blake *et. al.* (1994) to overcome the shortcomings cited in the previous section by simultaneously estimating A_0, A_1, B, and $\overline{\mathbf{X}}$. This estimation can be achieved by jointly maximising log likelihood over all the parameters:

$$L(\{\mathbf{X}_i\}\,|A_0, A_1, B, \overline{\mathbf{X}}) \quad = \quad -\frac{1}{2}\sum_{n=1}^{N} \left|B^{-1}\mathbf{R}_n\right|^2 - \log\det B,$$

where $\mathbf{R}_n$ is an instantaneous measure of error of fit:

$$\mathbf{R}_n \stackrel{\triangle}{=} \mathbf{X}_{n+2} - A_1\mathbf{X}_{n+1} - A_0\mathbf{X}_n - (I - A_0 - A_1)\overline{\mathbf{X}} \tag{5.19}$$

Solving explicitly for $\overline{\mathbf{X}}$ together with A_0, A_1 and B is a nonlinear problem, because L is not quadratic but quartic in the unknowns due to the product terms $A_0\mathbf{X}_n, A_1\mathbf{X}_{n+1}, A_0\overline{\mathbf{X}}$ and $A_1\overline{\mathbf{X}}$ that appear in $\mathbf{R}_n$. However, by defining a new, constant parameter

$$\mathbf{D} \stackrel{\triangle}{=} (I - A_0 - A_1)\overline{\mathbf{X}} \tag{5.20}$$

so that $\mathbf{R}_n$ becomes $\mathbf{X}_{n+2} - A_1\mathbf{X}_{n+1} - A_0\mathbf{X}_n + D$, the system is once again made linear.

The values of A_0, A_1, B and $\mathbf{D}$ that maximise the likelihood equation can be found by solving equations 5.21, 5.22, and 5.23. However, because the proof is involved and is a simpler case of the more general proof in section 5.7.2, it will be omitted for now.

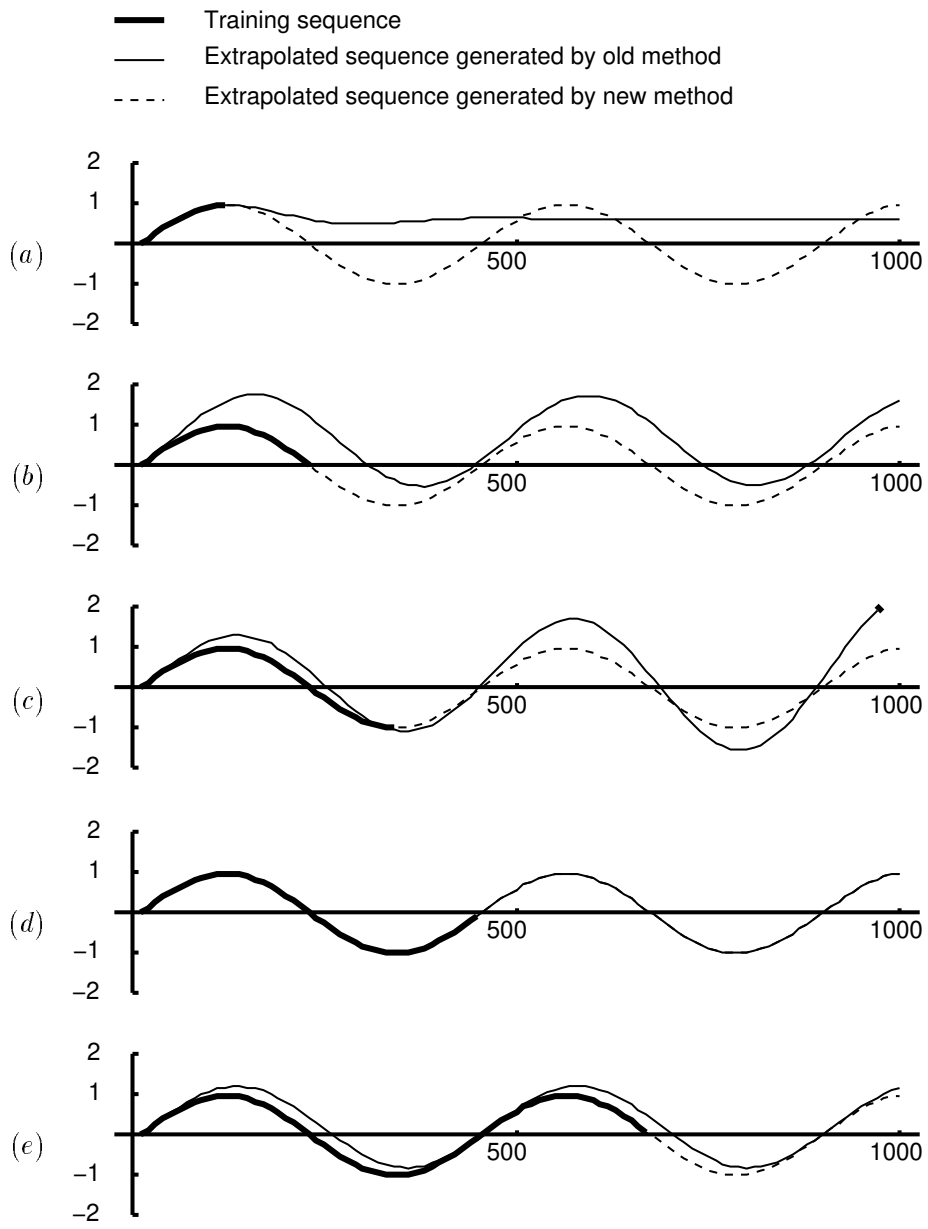


Figure 5.8: **Poor estimates of the mean cause large errors in other parameters.** *When learning dynamics, the system mean is normally estimated before the other parameters using the formula* $\hat{\overline{\mathbf{X}}} = (1/N)\sum_{i=0}^{N} X_i$. *This is a poor estimate when the training sequences are short compared to the system's periods; as a result, other parameters will be misestimated. The method described in section 5.4.2 simultaneously learns the mean and the other parameters. The formula of the training sequence is* $X_{n+2} = 1.9998X_{n+1} - X_n$ *(an undamped sine wave of period 446 centered at the origin with no added noise). In each graph the data used for learning are shown as a heavy line, the thin line is a new sequence generated from the dynamics learnt from the traditional method, and the dotted line is a sequence generated from dynamics learnt from the new method. The new method exactly learns the original model. The old method learns systems substantially different than the training sequence, but the longer the training sequence, the better the old method works.*

Simultaneously solving the following:

$$\mathbf{S}_2 - A_0\mathbf{S}_0 - A_1\mathbf{S}_1 - \mathbf{D}(N-2) = 0 \tag{5.21}$$

$$S_{20} - A_0 S_{00} - A_1 S_{10} = 0 \tag{5.22}$$

$$S_{21} - A_0 S_{01} - A_1 S_{11} = 0, \tag{5.23}$$

where $\mathbf{S}_i$ and S_{ij} are defined to be

$$\mathbf{S}_i = \sum_{n=1}^{N-2} \mathbf{X}_{n+i}, \quad i = 0, 1, 2, \tag{5.24}$$

$$S_{ij} = \sum_{n=1}^{N-2} \left(\mathbf{X}_{n+i}\mathbf{X}_{n+j}^T - \frac{1}{N-2}\mathbf{S}_i(\mathbf{S}_j)^T \right), \quad i, j = 0, 1, 2, \tag{5.25}$$

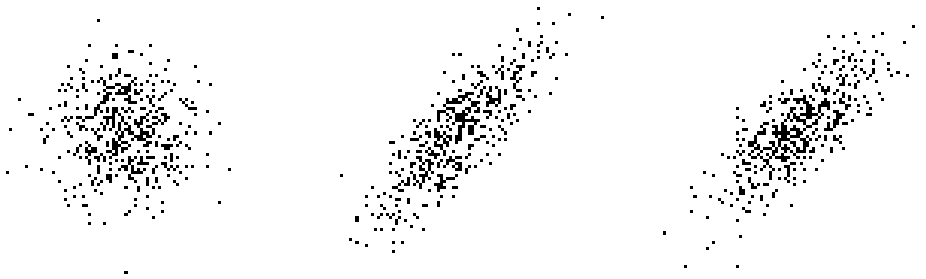
gives estimates of A_0, A_1 and $\mathbf{D}$. This leaves only the random process coupling parameter B to be determined. In fact, B cannot be determined uniquely (see figure 5.9 for a complete explanation), but any solution of $BB^T = C$ is acceptable, where

$$C = \frac{1}{N-2} \sum_{n=1}^{N-2} \mathbf{R}_n \mathbf{R}_n^T. \tag{5.26}$$

By substituting values of A_0, A_1 and $\mathbf{D}$ into equation 5.20, the mean of the sequence $\overline{\mathbf{X}}$ can be obtained explicitly, provided that $(I - A_0 - A_1)$ is non-singular. Singularity arises in certain special cases, for instance constant velocity dynamics, when estimated dynamical parameters are non-unique. However, in the case that the learnt dynamics will be used with a Kalman filter, it turns out that it is not necessary to recover $\overline{\mathbf{X}}$, only $\mathbf{D}$ (the Kalman filter uses equation 5.6, not equation 5.5). Singularities will be discussed more completely in section 5.7.5.

5.5 Learning dynamics from multiple training sequences

Blake, *et. al.* show that it is possible to learn complex motions by combining several training sequences. Each training sequence contains an example of a specific motion, such as a rotation or a particular scaling. The likelihood equation can then be maximised to find the



(a) 500 samples of $\mathbf{w}_i$ (b) 500 samples of $B\mathbf{w}_i$ (c) 500 samples of $B^*\mathbf{w}_i$

Figure 5.9: **Why B can't be uniquely determined.** *Learning is the process of observing a system, then using those observations to recover the variables A_0, A_1, and B from equation 5.5. A_0 and A_1 describe the deterministic part of the equations: they specify how the system behaves in the absence of noise. $B\mathbf{w}_i$ is the stochastic part of the equation. $\mathbf{w}_i$ is a sample from a vector of N independent unit Gaussian variables, and B "colours" that sample so that $B\mathbf{w}_i$ has the desired covariance C. Figure 5.9a shows 500 samples from a vector of 2 independent unit Gaussians. Figure 5.9b shows 500 more samples colored by premultiplying the matrix $B = \left(\begin{smallmatrix} 1 & 0.5 \\ 0.5 & 1 \end{smallmatrix}\right)$. Since B scales and introduces dependencies between the components of $\mathbf{w}_i$, the cloud has been reshaped into an ellipse elongated along the axis $x = y$, and the covariance of this sample is $C = BB^T$. However, B is not unique. Figure 5.9c shows 500 more samples, this time colored by $B^* = B\left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$. B^* first rotates the samples by 90°, then shapes them as before with B. Since the cloud has no orientation, the net effect of the B and B^* is the same: a cloud with covariance $C = B^*(B^*)^T = BRR^TB^T = BB^T$, where R is the 90° rotation matrix (or indeed any rotation matrix).*

single best model that fits all the training sequences, and the learnt model will then exhibit the motions from all of the training sequences.

Merely concatenating the multiple sequences into one long sequence is not sufficient, because the system will learn to compensate for the discontinuity from one sequence to the next.

In the case when the mean of all the training sequences was assumed to be zero (Blake et al., 1995), the extension to the learning algorithm to handle multiple sequences was trivial. Fortunately, when $\overline{\mathbf{X}}$ is being simultaneously estimated with A_0, A_1 and B, the extension is still straightforward. Given sequences numbered $\gamma = 1, \ldots, \Gamma$, with the nth data point of the γth training sequence being represented by $\mathbf{X}_n^\gamma$, and the γth data sequence

having N_γ data points, the S-moments from the γth data sequence are defined to be

$$\begin{aligned} \mathbf{S}_i^\gamma &= \sum_{n=1}^{N_\gamma - 2} \mathbf{X}_{n+i}^\gamma, \quad i = 0, 1, 2, \\ S_{ij}^\gamma &= \sum_{n=1}^{N_\gamma - 2} \mathbf{X}_{n+i}^\gamma (\mathbf{X}_{n+j}^\gamma)^T, \quad i, j = 0, 1, 2. \end{aligned} \tag{5.27}$$

Because all of the training sequences are assumed to have exactly the same dynamical model (including the same $\overline{\mathbf{X}}$), the moments describing the cumulative system can be written as the sum of the moments, correcting for the effect of $\overline{\mathbf{X}}$ on the S_{ij} moments:

$$\begin{aligned} \mathbf{S}_i &= \sum_{\gamma=1}^{\Gamma} \mathbf{S}_i^\gamma \\ S_{ij} &= -\mathbf{S}_i \mathbf{S}_j^T + \sum_{\gamma=1}^{\Gamma} S_{ij}^\gamma \end{aligned} \tag{5.28}$$

Using this augmented definition of moments, the log-likelihood equation can be minimised by equations 5.21, 5.22, and 5.23. Once again the proof is omitted because it is subsumed by the proof detailed in section 5.7.2.

5.6 Shape and motion variability

The second problem addressed by this chapter concerns learning dynamical models which represent both *class* variability and *dynamical* variability. Dynamical variability describes changes that occur due to the motion of a single object. These changes can either be projective effects resulting from the rigid movement of an object relative to the camera, from changes due to self-occlusion or occlusion by another object, or from changes in the object itself, such as those resulting from articulation. Class variability describes the differences between objects that are similar to each other but not identical. In the class of human faces, each face is unique and identifiable, but all faces share many common characteristics. Variability of this kind can be learned effectively by Principal Components Analysis applied to a linear curve parameterisation (Cootes et al., 1993).

Previous systems for learning dynamics have failed to address the important distinction between the two sources of variability by learning dynamical variability without considering

class variability (Pentland and Horowitz, 1991; Blake and Isard, 1994; Baumberg and Hogg, 1995b), or grouping both kinds of variability together (Ferrier et al., 1994; Cootes et al., 1995). If the two sources of variability are lumped together into one dynamical model, there will be insufficient constraint on motion variability. A tracker incorporating such a model as a predictor will allow temporal motion/shape changes to range over the *total* modelled variability, both class and temporal. This is inappropriate. Rather, modelled class variability should apply only as a tracker is initialised on a new object. Once tracking is underway, the identity of the object does not change, so class variability should be suspended. Instead, the dynamical variability model should take over, relatively tightly constrained as it allows only the variability normally associated with motion.

Maintaining the distinction between the two sources of variability is of considerable practical importance. In a robotic application, a mobile tractor-robot monitors the motion of plants from a downward pointing camera, in order to control the application of sprayed chemicals from a spray-nozzle array (figure 1.1). Moving plants in the video stream are tracked by a dynamic contour tracker (figure 1.2).

A dynamical model was learnt from video sequences of more than 30 plants. Class and motion variability were modelled jointly as a stochastic process. Inter-class variability proves to be considerably greater than motion variability in this kind of data. Figure 5.10a shows the shape of five plants as approximated by the Active Contour. Variation between plants is substantial. Figure 5.10b shows the shape of the same plant as approximated by the Active Contour at five different times in the tracking sequence. Shape variation for a single plant is very small, especially when compared with figure 5.10a.

When this model was used in a Kalman filter tracker, there was excessive allowance for variability over time, resulting in sporadic inaccuracies in tracked shape. Figure 5.11 shows how jointly modelling inter-plant and intra-plant differences can cause Active Contours to fail even in the case of synthetic data with no added noise or distracting features. Such systems incorrectly estimate the means, which causes prediction to be consistently biased

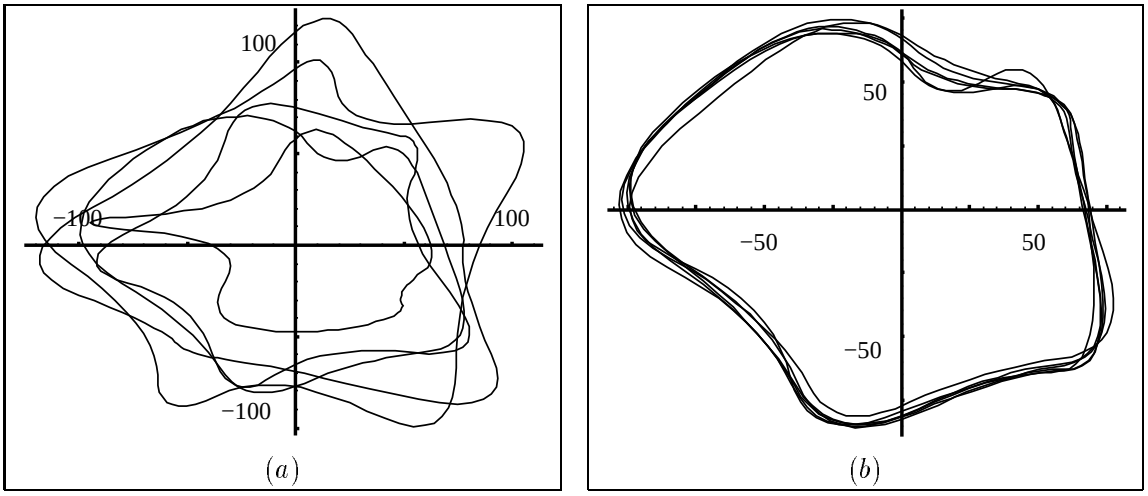


Figure 5.10: **Class-variability swamps motion-variability.** *(a) Measured class-variation over five plants is considerable. In contrast (b) variation due to the motion of one individual plant, as it passes through the camera's field of view, is relatively small.*

towards the combined mean of all the sequences (see figure 5.6).

Figures 5.16 and 5.17 compare the tracking results of Active Contours using unlearnt dynamics, learnt dynamics that jointly model inter-plant and intra-plant differences, and learnt dynamics that model inter-plant and intra-plant differences independently. On an easier sequence (figure 5.16), learnt dynamics are shown to be superior to unlearnt dynamics. On a more complicated sequence (figure 5.17), the excessive variability of the joint model causes the Active Contour to fail when the independent model succeeds. As with synthetic results, joint-modelling systems have biased prediction which can cause tracking to fail (see figure 5.6).

5.7 Modelling class and motion variability

The distinction between class and motion variability is important. In this section, multidimensional, second order dynamical models will be extended to model these two sources of variability independently. The resulting, augmented model, once installed as the predictor for a tracker, performs much more robustly than when all variability is modelled jointly.

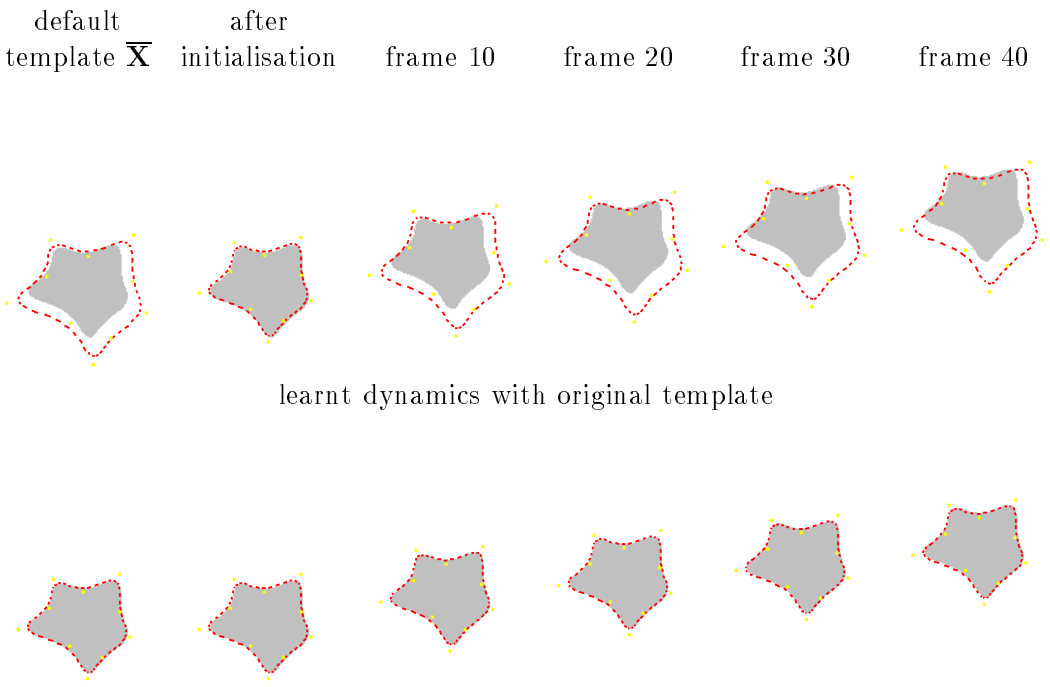


Figure 5.11: **Using the wrong template ruins tracking with learnt dynamics** *To simulate the tracking of a small plant, a synthetic tracking sequence is generated of a plant twenty percent smaller than the template being used. During the sequence, the plant translates vertically but scales little. The sequence is tracked using a tracker with default dynamics and template. From this tracking the dynamics are learnt using equations 5.21 to 5.23. The top row shows tracking using the learnt dynamics and the original template. Because there was no scaling in the training sequence, the learnt dynamics are intolerant to scaling. When presented with the same training sequence (and its twenty percent smaller plants) tracking fails quickly in spite of perfect initialisation. The bottom row shows that once the template is adjusted to compensate for the smaller plant, tracking is perfect.*

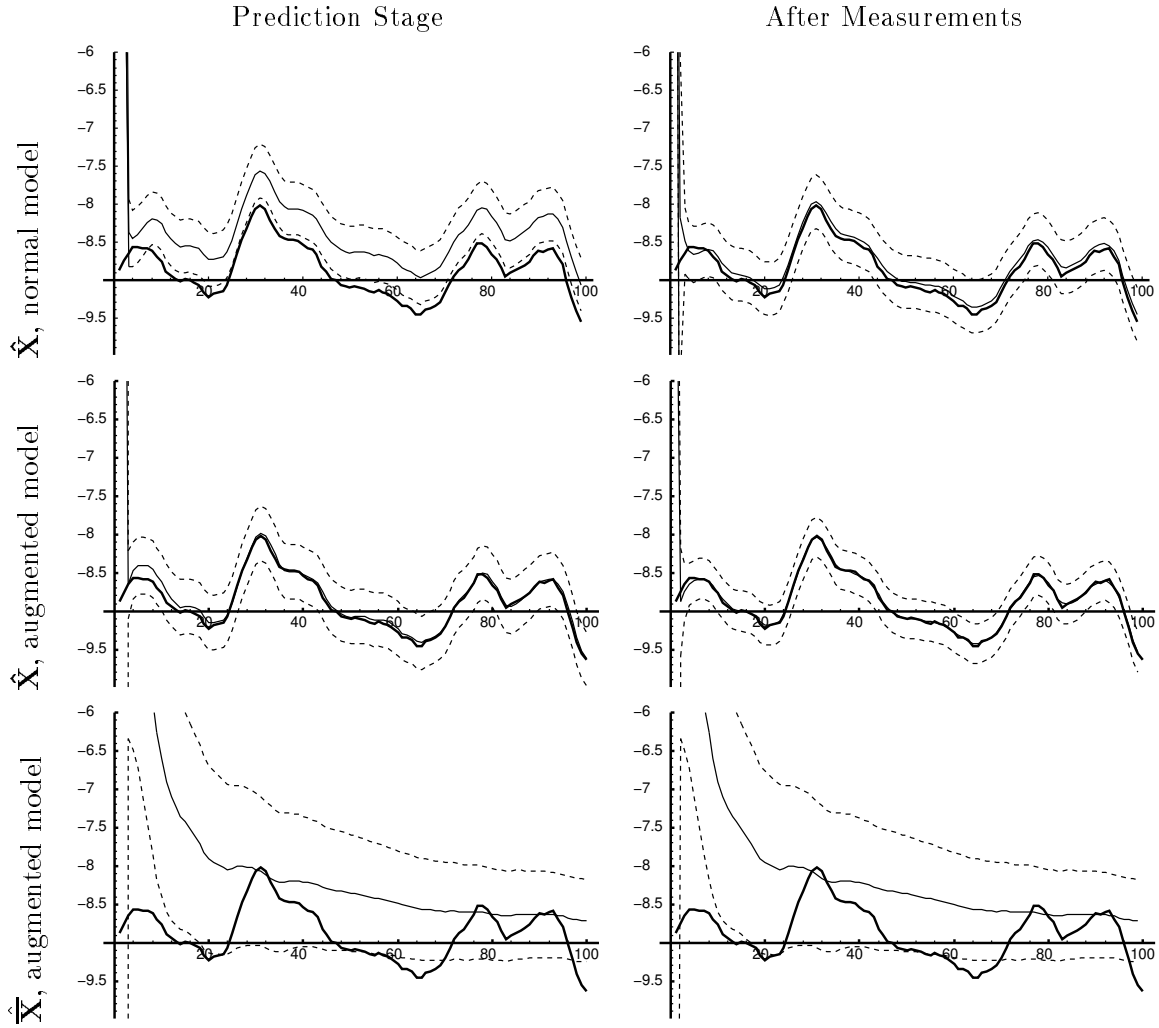


Figure 5.12: **Mean learning dynamics outperform normal dynamics when class variability swamps motion-variability.** *The heavy, solid line represents the measurements, the thin, solid line represents the state estimates and the dotted lines show two standard deviations of the estimated error. Six synthetic zero-mean training sequences were used to learn dynamics using both the normal model and the augmented model. Both models were then used to track one of the training sequences. In the sequences the variation of motion about the mean (motion-variability) is much smaller than the variation of the mean itself (class-variability). The normal motion model doesn't model the two variations separately. As a consequence, prediction is always strongly biased. A tracker using a range-gate would probably fail as a result of the poor prediction. Additionally, the correction applied at each step is very large. The augmented model models class and motion variability separately and has superior prediction and correction. Though the initial estimate of* $\hat{\overline{\mathbf{X}}}$ *is poor the system converges to the mean of the sequence.*

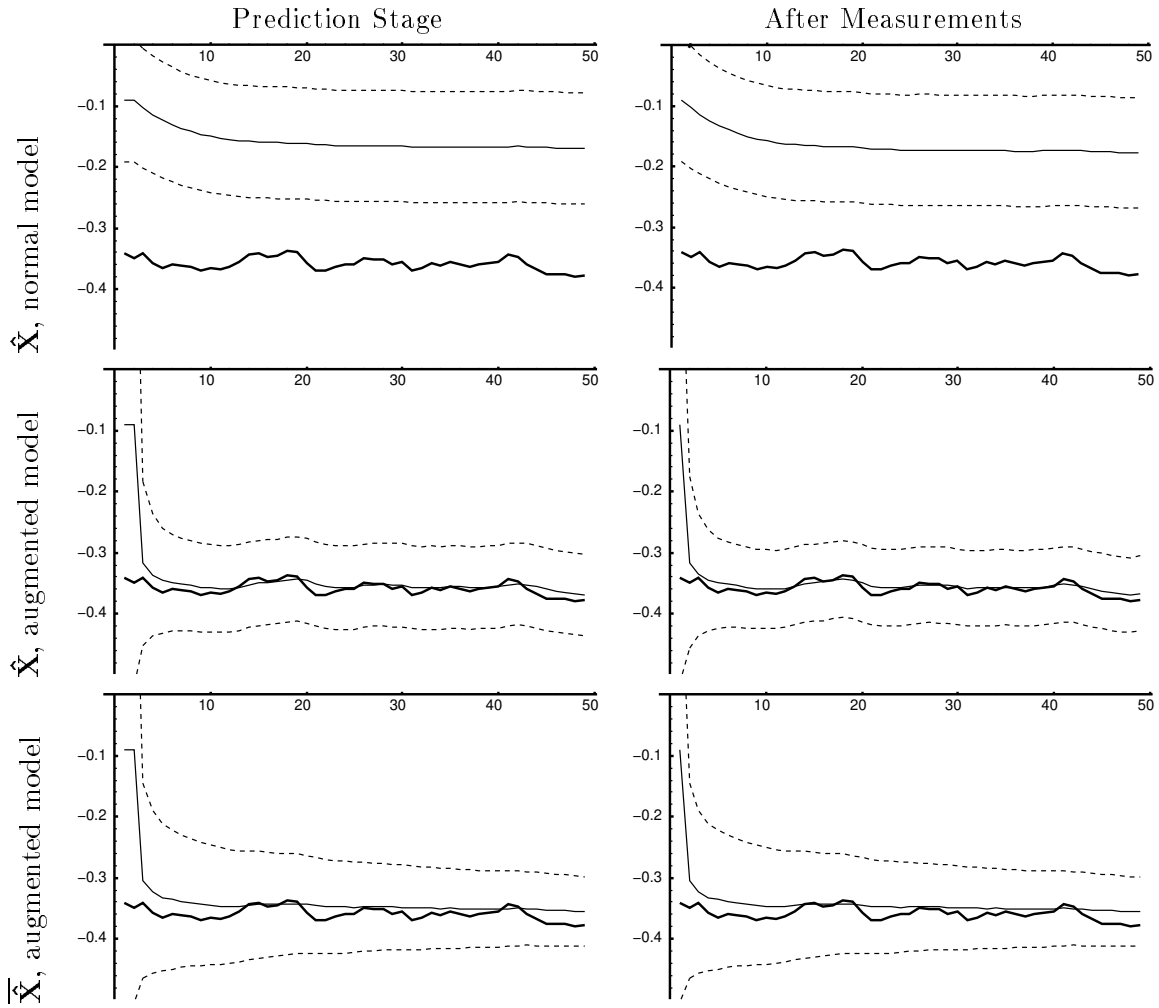


Figure 5.13: **Augmented dynamics perform better on real data sequences** *The fourth affine-dimension was extracted from six sequences of tracking of cauliflowers. The six sequences were then used to learn dynamics using the normal and augmented models. Because the motion variability relative to the sequence means was very small compared to the variance of the means, the normal model has a strong steady state prediction error. This steady state error causes problems when Kalman filters with range gates are used. The augmented model models the variation separately and does not exhibit steady state error either in the prediction stage or after measurements. The estimate of* $\overline{\hat{\mathbf{X}}}$ *converges very quickly and does not follow the valley in the data at frame 42. Note that while the uncertainty of the state doesn't decrease from frame 10 on, the uncertainty of the mean is constantly decreasing.*

5.7.1 Extended dynamical model

The improved model extends the state vector by regarding the mean configuration $\overline{\mathbf{X}}$ as a variable, rather than a constant as above. Now the dynamical model, most conveniently written in block-matrix form, is:

$$\begin{pmatrix} \mathbf{X}_{n+1} \\ \mathbf{X}_{n+2} \\ \overline{\mathbf{X}}_{n+2} \end{pmatrix} = \begin{pmatrix} 0 & I & 0 \\ A_0 & A_1 & I - A_0 - A_1 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{X}_{n+1} \\ \overline{\mathbf{X}}_{n+1} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ B\mathbf{w}_n \\ \mathbf{0} \end{pmatrix} \tag{5.29}$$

This simply repeats the original model (5.6) but with the addition of an equation that states that $\overline{\mathbf{X}}$ is constant over time. It might seem that this is identical to the old model. The crucial difference is that $\overline{\mathbf{X}}$ is not known *a priori*, but is estimated online. Once this dynamical system is installed as the predictor for a Kalman filter tracker, $\overline{\mathbf{X}}$ is initialised with its *estimated* mean value $\overline{\mathbf{X}_0}$ and associated variances and covariances. The initial estimate of the mean ($\mathsf{E}\left(\overline{\mathbf{X}}\right)$) and covariance ($\mathsf{Var}\left(\overline{\mathbf{X}}\right)$) are obtained by statistical estimation from training data (see equations 5.48 and 5.49)[2]. The value of $\overline{\mathbf{X}}_n$ converges rapidly in the Kalman filter as the mean shape of the object currently being tracked is estimated. Once estimated, the mean shape remains fixed, reinitialised only when a new object is to be tracked.

5.7.2 Learning algorithm

One of the main new theoretical results presented is the learning algorithm for the extended system dynamics. Given objects labelled $\gamma = 1, \ldots, \Gamma$, each observed in motion as data sequences $\mathbf{X}_n^\gamma$, $\gamma = 1, \ldots, \Gamma$, $n = 1, \ldots, N_\gamma$ the problem is to estimate global parameters for the dynamics: A_0, A_1, B and mean configurations for each object $\overline{\mathbf{X}}^\gamma$ by jointly maximising log likelihood over all the parameters:

$$\begin{aligned} L(\{\mathbf{X}_n^\gamma\} | A_0, A_1, B, \overline{\mathbf{X}}^1, \ldots, \overline{\mathbf{X}}^\Gamma) = & - \frac{1}{2} \sum_{\gamma=1}^{\Gamma} \sum_{n=1}^{N_\gamma} \left| B^{-1} \mathbf{R}_n^\gamma \right|^2 \\ & - \left(\sum_\gamma (N_\gamma - 2) \right) \log \det B, \end{aligned} \tag{5.30}$$

[2]The problem of choosing a correct initial covariance will be discussed in greater detail in chapter 7

where $\mathbf{R}_n^\gamma$ is an instantaneous measure of error of fit:

$$\mathbf{R}_n^\gamma \triangleq \mathbf{X}_{n+2}^\gamma - A_1\mathbf{X}_{n+1}^\gamma - A_0\mathbf{X}_n^\gamma - (I - A_0 - A_1)\overline{\mathbf{X}}^\gamma. \tag{5.31}$$

Solving explicitly for the $\overline{\mathbf{X}}^\gamma$ together with A_0, A_1, B is a nonlinear problem, because L is not quadratic but quartic in the unknowns, due to the product terms $A_0\overline{\mathbf{X}}^\gamma$ and $A_1\overline{\mathbf{X}}^\gamma$ that appear in $\mathbf{R}_n^\gamma$. The nonlinearity can be removed in a similar way to section 5.4.2 by defining a constant parameter $\mathbf{D}^\gamma$:

$$\mathbf{D}^\gamma = (I - A_0 - A_1)\overline{\mathbf{X}}^\gamma \tag{5.32}$$

so that the definition of $\mathbf{R}_n^\gamma$ becomes

$$\mathbf{R}_n^\gamma = \mathbf{X}_{n+2}^\gamma - A_1\mathbf{X}_{n+1}^\gamma - A_0\mathbf{X}_n^\gamma - \mathbf{D}^\gamma. \tag{5.33}$$

While it is not obvious, it turns out that it is possible to maximise the likelihood function in parts. Maximising first with respect to the deterministic parameters $(A_0, A_1, \overline{\mathbf{X}}^1, \ldots \overline{\mathbf{X}}^\Gamma)$, it will be shown later that this solution is independent of the stochastic parameter B.

Equivalent to maximising L we minimise

$$\begin{aligned} f(A_0, A_1, \overline{\mathbf{X}}^\gamma, \ldots, \overline{\mathbf{X}}^\gamma) &= \sum_{\gamma=1}^{\Gamma}\sum_{n=1}^{N_\gamma} \left|B^{-1}\mathbf{R}_n^\gamma\right|^2 \qquad (5.34) \\ &= \sum_{\gamma=1}^{\Gamma}\sum_{n=1}^{N_\gamma} (\mathbf{R}_n^\gamma)^T B^{-T}B^{-1}\mathbf{R}_n^\gamma \\ &= \sum_{\gamma=1}^{\Gamma}\sum_{n=1}^{N_\gamma} \mathrm{tr}\left((\mathbf{R}_n^\gamma)^T B^{-T}B^{-1}\mathbf{R}_n^\gamma\right) \\ &= \sum_{\gamma=1}^{\Gamma}\sum_{n=1}^{N_\gamma} \mathrm{tr}\left(\mathbf{R}_n^\gamma (\mathbf{R}_n^\gamma)^T B^{-T}B^{-1}\right) \\ &= \sum_{\gamma=1}^{\Gamma} \mathrm{tr}\left(\sum_{n=1}^{N_\gamma} \mathbf{R}_n^\gamma (\mathbf{R}_n^\gamma)^T C^{-1}\right) \qquad (5.35) \end{aligned}$$

where C is defined to be BB^T.

Now to maximise the likelihood we can differentiate equation 5.35 with respect to each of the variables. Because the likelihood function is the trace of matrix operation, it is possible

to use the lemmas in section A.3. First, expand the product

$$\sum_{\gamma=1}^{\Gamma} \mathrm{tr}\left(\sum_{n=1}^{N_\gamma} \mathbf{R}_n^\gamma (\mathbf{R}_n^\gamma)^T C^{-1}\right) = \sum_{\gamma=1}^{\Gamma} \mathrm{tr}\Big[C^{-1} \quad (T_{22}^\gamma - T_{21}^\gamma A_1^T - T_{20}^\gamma A_0^T - \mathbf{S}_2^\gamma (\mathbf{D}^\gamma)^T$$
$$- A_1 T_{12}^\gamma + A_1 T_{11}^\gamma A_1^T + A_1 T_{10}^\gamma A_0^T + A_1 \mathbf{S}_1^\gamma (\mathbf{D}^\gamma)^T$$
$$- A_0 T_{02}^\gamma + A_0 T_{01}^\gamma A_1^T + A_0 T_{00}^\gamma A_0^T + A_0 \mathbf{S}_0^\gamma (\mathbf{D}^\gamma)^T$$
$$- \mathbf{D}^\gamma (\mathbf{S}_2^\gamma)^T + \mathbf{D}^\gamma (\mathbf{S}_1^\gamma)^T A_1^T + \mathbf{D}^\gamma (\mathbf{S}_0^\gamma)^T A_0^T$$
$$+ (N_\gamma - 2)\mathbf{D}^\gamma (\mathbf{D}^\gamma)^T)\Big] \tag{5.36}$$

where T_{ij}^γ is defined as

$$T_{ij}^\gamma = \sum_{k=1}^{N_\gamma} \mathbf{X}_{k+i}^\gamma (\mathbf{X}_{k+j}^\gamma)^T. \tag{5.37}$$

Now, maximise it with respect to $\mathbf{D}^\gamma$ by taking the derivative with respect to $\mathbf{D}^\gamma$:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{D}^\gamma} \sum_{g=1}^{\Gamma} \mathrm{tr}\left(C^{-1} \sum_{n=1}^{N_g} \mathbf{R}_n^g (\mathbf{R}_n^g)^T\right) &= \sum_{g=1}^{\Gamma} \mathrm{tr}\left(C^{-1} \sum_{n=1}^{N_\gamma} \frac{\partial}{\partial \mathbf{D}^\gamma} \mathbf{R}_n^g (\mathbf{R}_n^g)^T\right) \\
&= \delta_{g\gamma} \sum_{g=1}^{\Gamma} \mathrm{tr}\left(C^{-1} \sum_{n=1}^{N_\gamma} \frac{\partial}{\partial \mathbf{D}^\gamma} \mathbf{R}_n^g (\mathbf{R}_n^g)^T\right) \\
&= \mathrm{tr}\left(C^{-1} \sum_{n=1}^{N_\gamma} \frac{\partial}{\partial \mathbf{D}^\gamma} \mathbf{R}_n^\gamma (\mathbf{R}_n^\gamma)^T\right) \\
&= \mathrm{tr}\Big[C^{-1} \frac{\partial}{\partial \mathbf{D}^\gamma} (-\mathbf{S}_2^\gamma (\mathbf{D}^\gamma)^T + A_1 \mathbf{S}_1^\gamma (\mathbf{D}^\gamma)^T + A_0 \mathbf{S}_0^\gamma (\mathbf{D}^\gamma)^T \\
&\quad - \mathbf{D}^\gamma (\mathbf{S}_2^\gamma)^T + \mathbf{D}^\gamma (\mathbf{S}_1^\gamma)^T A_1^T + \mathbf{D}^\gamma (\mathbf{S}_0^\gamma)^T A_0^T \\
&\quad + (N_\gamma - 2)\mathbf{D}^\gamma (\mathbf{D}^\gamma)^T)\Big] \\
&= \mathrm{tr}\Big[C^{-1} \frac{\partial}{\partial \mathbf{D}^\gamma} (-\mathbf{S}_2^\gamma (\mathbf{D}^\gamma)^T - \mathbf{D}^\gamma (\mathbf{S}_2^\gamma)^T + A_1 \mathbf{S}_1^\gamma (\mathbf{D}^\gamma)^T \\
&\quad + \mathbf{D}^\gamma (\mathbf{S}_1^\gamma)^T A_1^T + A_0 \mathbf{S}_0^\gamma (\mathbf{D}^\gamma)^T + \mathbf{D}^\gamma (\mathbf{S}_0^\gamma)^T A_0^T \\
&\quad + (N_\gamma - 2)\mathbf{D}^\gamma (\mathbf{D}^\gamma)^T)\Big] \\
&= \mathrm{tr}\Big[C^{-1} \frac{\partial}{\partial \mathbf{D}^\gamma} (-2\mathbf{D}^\gamma (\mathbf{S}_2^\gamma)^T + 2\mathbf{D}^\gamma (\mathbf{S}_1^\gamma)^T A_1^T \\
&\quad + 2\mathbf{D}^\gamma (\mathbf{S}_0^\gamma)^T A_0^T + (N_\gamma - 2)\mathbf{D}^\gamma (\mathbf{D}^\gamma)^T)\Big] \\
&= \mathrm{tr}\Big[C^{-1} (-2\mathbf{S}_2^\gamma + 2A_1 \mathbf{S}_1^\gamma + 2A_0 \mathbf{S}_0^\gamma + 2(N_\gamma - 2)\mathbf{D}^\gamma)\Big].
\end{aligned}$$

This results in a set of Γ simultaneous equations to solve, all of which are a function of A_0 and A_1.

Similarly, the likelihood can be maximised in terms of A_0 and A_1:

$$\begin{aligned}
\frac{\partial}{\partial A_0}\sum_{\gamma=1}^{\Gamma}\mathrm{tr}\left(\sum_{n=1}^{N_\gamma}\mathbf{R}_n^\gamma\left(\mathbf{R}_n^\gamma\right)^T C^{-1}\right) &= \frac{\partial}{\partial A_0}\mathrm{tr}\left(\sum_{\gamma=1}^{\Gamma}\sum_{n=1}^{N_\gamma} C^{-1}\mathbf{R}_n^\gamma\left(\mathbf{R}_n^\gamma\right)^T\right)\\
&= \frac{\partial}{\partial A_0}\mathrm{tr}\Big[\sum_{\gamma=1}^{\Gamma} C^{-1}\Big(- T_{20}^\gamma A_0^T + A_1 T_{10}^\gamma A_0^T - A_0 T_{02}^\gamma\\
&\quad + A_0 T_{01}^\gamma A_1^T + A_0 T_{00}^\gamma A_0^T + A_0\mathbf{S}_0^\gamma(\mathbf{D}^\gamma)^T + \mathbf{D}^\gamma(\mathbf{S}_0^\gamma)^T A_0^T\Big)\Big]\\
&= \mathrm{tr}\Big[\sum_{\gamma=1}^{\Gamma} C^{-1}\left(-T_{20}^\gamma + A_1 T_{10}^\gamma + A_0 T_{00}^\gamma + \mathbf{D}^\gamma(\mathbf{S}_0^\gamma)^T\right)\\
&\quad + C^{-T}\left(-(T_{02}^\gamma)^T + A_1(T_{01}^\gamma)^T + A_0(T_{00}^\gamma)^T + \mathbf{D}^\gamma(\mathbf{S}_0^\gamma)^T\right)\Big]\\
&= \mathrm{tr}\Big[\sum_{\gamma=1}^{\Gamma} 2C^{-1}\left(-T_{20}^\gamma + A_1 T_{10}^\gamma + A_0 T_{00}^\gamma + \mathbf{D}^\gamma(\mathbf{S}_0^\gamma)^T\right)\Big]\\
&= \mathrm{tr}\Big[2C^{-1}\left(-S_{20} + A_1 S_{10} + A_0 S_{00}\right)\Big], \qquad (5.38)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial A_1}\sum_{\gamma=1}^{\Gamma}\mathrm{tr}\left(\sum_{n=1}^{N_\gamma}\mathbf{R}_n^\gamma\left(\mathbf{R}_n^\gamma\right)^T C^{-1}\right) &= \frac{\partial}{\partial A_1}\mathrm{tr}\left(\sum_{\gamma=1}^{\Gamma}\sum_{n=1}^{N_\gamma} C^{-1}\mathbf{R}_n^\gamma\left(\mathbf{R}_n^\gamma\right)^T\right)\\
&= \frac{\partial}{\partial A_1}\mathrm{tr}\sum_{\gamma=1}^{\Gamma}\Big[C^{-1}\Big(- T_{21}^\gamma A_1^T - A_1 T_{12}^\gamma + A_1 T_{11}^\gamma A_1^T\\
&\quad + A_1 T_{10}^\gamma A_0^T + A_1\mathbf{S}_1^\gamma(\mathbf{D}^\gamma)^T + A_0 T_{01}^\gamma A_1^T + \mathbf{D}^\gamma(\mathbf{S}_1^\gamma)^T A_1^T\Big)\Big]\\
&= \mathrm{tr}\Big[\sum_{\gamma=1}^{\Gamma} C^{-1}\left(-T_{21}^\gamma + A_1 T_{11}^\gamma + A_0 T_{01}^\gamma + \mathbf{D}^\gamma(\mathbf{S}_1^\gamma)^T\right)\\
&\quad + C^{-T}\left(-(T_{12}^\gamma)^T + A_1(T_{11}^\gamma)^T + A_0(T_{10}^\gamma)^T + \mathbf{D}^\gamma(\mathbf{S}_1^\gamma)^T\right)\Big]\\
&= \mathrm{tr}\Big[\sum_{\gamma=1}^{\Gamma} 2C^{-1}\left(-T_{21}^\gamma + A_1 T_{11}^\gamma + A_0 T_{01}^\gamma + \mathbf{D}^\gamma(\mathbf{S}_1^\gamma)^T\right)\Big]\\
&= \mathrm{tr}\Big[2C^{-1}\left(-S_{21} + A_1 S_{11} + A_0 S_{01}\right)\Big]. \qquad (5.39)
\end{aligned}$$

where $\mathbf{S}_i, S_{ij}$ are certain moments of the data-set, defined in equation 5.46. Thus, the following set of $\Gamma + 2$ equations is solved for the estimated values of parameters A_0, A_1 and

$\mathbf{D}^\gamma,\ \gamma = 1, \ldots, \Gamma$:

$$\mathbf{S}_2^\gamma - A_0\mathbf{S}_0^\gamma - A_1\mathbf{S}_1^\gamma - D^\gamma(N_\gamma - 2) = 0, \quad \gamma = 1, \ldots, \Gamma, \tag{5.40}$$

$$S_{20} - A_0 S_{00} - A_1 S_{10} = 0 \tag{5.41}$$

$$S_{21} - A_0 S_{01} - A_1 S_{11} = 0. \tag{5.42}$$

Fortunately, the system can be solved in parts to reduce the computational complexity. The parameters A_0 and A_1 can be recovered from simultaneously solving equations 5.41 and 5.42. Then $\mathbf{D}^1$ can be recovered from solving equation 5.40 with $\gamma = 1$, $\mathbf{D}^2$ can be recovered from solving equation 5.40 with $\gamma = 2$, and so on.

This leaves only the random process coupling parameter B to be determined. In fact B cannot be determined uniquely (see figure 5.9), but any solution of $BB^T = C$ is acceptable (i.e. B is the matrix square root of C), where

$$C = \frac{1}{\sum_\gamma (N_\gamma - 2)} \sum_\gamma Z^\gamma(A_0, A_1, \mathbf{D}^\gamma). \tag{5.43}$$

and

$$Z^\gamma(A_0, A_1, \mathbf{D}^\gamma) = \sum_{n=1}^{N_\gamma} \mathbf{R}_n^\gamma \left(\mathbf{R}_n^\gamma\right)^T. \tag{5.44}$$

It remains to define the moments of the data-set[3]:

$$S_{ij}^\gamma = \sum_{n=1}^{N_\gamma - 2} \mathbf{X}_{n+i}^\gamma (\mathbf{X}_{n+j}^\gamma)^T - \sum_\gamma \frac{1}{N_\gamma - 2} \mathbf{S}_i^\gamma (\mathbf{S}_j^\gamma)^T, \quad i, j = 0, 1, 2, \tag{5.45}$$

and

$$S_{ij} = \sum_\gamma S_{ij}^\gamma. \tag{5.46}$$

Similarly,

$$\mathbf{S}_i^\gamma = \sum_{n=1}^{N_\gamma - 2} \mathbf{X}_{n+i}^\gamma, \quad i = 0, 1, 2. \tag{5.47}$$

[3]Moments defined here are quite different from the moments used in (Blake and Isard, 1994; Baumberg and Hogg, 1995b) despite the similarity of the S_{ij} notation.

5.7.3 Recovery of $\overline{\mathbf{X}}^{\gamma}$

Object means $\overline{\mathbf{X}}^{\gamma}$ can be obtained explicitly, provided $(I - A_0 - A_1)$ is non-singular, by solving for them in equation 5.32. (The problem of singular matrices is discussed in section 5.7.5.) The mean and covariance of the set of object-means $\overline{\mathbf{X}}^{\gamma}, \quad \gamma = 1, \ldots, \Gamma$ are computed to use as initial mean and covariance for the additional state variable $\overline{\mathbf{X}}$ (mean-shape) in the tracking procedure:

$$\overline{\mathbf{X}}_0 = \frac{1}{\Gamma}\sum_{\gamma=1}^{\Gamma}\overline{\mathbf{X}}^{\gamma} \tag{5.48}$$

$$\mathsf{Var}(\overline{\mathbf{X}}_0) = \frac{1}{(\Gamma-1)}\sum_{\gamma=1}^{\Gamma}(\overline{\mathbf{X}}^{\gamma} - \overline{\mathbf{X}}_0)(\overline{\mathbf{X}}^{\gamma} - \overline{\mathbf{X}}_0)^T \tag{5.49}$$

Once all the parameters have been learned, the tracker has greatly improved performance when compared to the tracker that jointly modeled inter-plant and intra-plant variations. The tracking parameters are learnt correctly even in the case of short training sequences (figure 5.8). The prediction stage of the filter is no longer biased (figures 5.6 and 5.6). Most importantly, the Active Contour tracker is capable of tracking more robustly in the presence of noise (figure 5.17).

5.7.4 Observability and controllability of the augmented state equations

In the previous sections, it was observed that in some cases, it is not possible to learn all the means of the systems, which results in some of the system matrices being singular. This drop in rank has well-known effects on the performance of the Kalman filter, which can most easily be explained by introducing the concepts of *observability* and *controllability*.

When using a Kalman filter for the tracking problem, it is important that there be an accurate model of the expected shape and motion. Additionally, there are other conditions that are necessary for the Kalman filter to work correctly. Firstly, there must exist a matrix H relating the state the filter is estimating to the measurements the filter has made. Secondly, it must be possible to measure all the states of the filter through the measurements. This is called observability. When a system is unobservable it means that

there are states that the Kalman filter cannot learn anything about. Thirdly, the noise matrix B must be able to insert noise into all of the states. This is called controllability. An uncontrollable system is one in which the uncertainty of one or more of the state's dimension falls to zero. In practice this usually means that the filter has more dimensions than are necessary.

The matrix H defines the relationship between the state and the measurements. Specifically,

$$\mathbf{Z}_n = H\mathbf{X}_n + \mu_n \tag{5.50}$$

where $\mathbf{Z}_n$ is the measurement at time n and μ_n is the (Gaussian) measurement noise at time n. In general $\mathbf{Z}$ will have a different number of dimensions than $\mathbf{X}$, and the matrix H may or may not have a special form. For illustrative purposes, it will be assumed that the measurement process measures in the state space, i.e. $\mathbf{Z}_n = \mathbf{X}_n + \mu_n$, though the same results apply with other measurement processes.

The observability of the model of a Kalman filter can be computed by examining the observability matrix $\mathcal{O}$ which is defined as:

$$\mathcal{O} \triangleq \begin{pmatrix} H \\ HF \\ HF^2 \\ \vdots \\ HF^{n-1} \end{pmatrix}, \tag{5.51}$$

where F is defined in equation 5.8. The system is observable if the matrix is of full rank, and it is unobservable if it is not. The H matrix of the fixed-mean model (equation 5.6) is of the form $\begin{pmatrix} I & 0 \end{pmatrix}$ when the measurement process measures the state space. It's observability matrix is:

$$\mathcal{O} = \begin{pmatrix} H \\ HF \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \tag{5.52}$$

which is clearly of full rank. Similarly, the observability of the augmented system is

$$\mathcal{O} \triangleq \begin{pmatrix} H \\ HF \\ HF^2 \end{pmatrix} \tag{5.53}$$

$$= \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ A_0 & A_1 & \Psi \end{pmatrix} \tag{5.54}$$

where $\Psi \triangleq I - A_0 - A_1$ has been defined for convenience. In this case, it is clear the system is observable if and only if Ψ is of full rank (i.e. Ψ is invertible).

Similarly, the controllability $\mathcal{C}$ matrix of a system is defined as

$$\mathcal{C} \triangleq \begin{pmatrix} \mathcal{B} & F\mathcal{B} & F^2\mathcal{B} & \cdots & F^{n-1}\mathcal{B} \end{pmatrix}. \tag{5.55}$$

Where $\mathcal{B}$ is the matrix coupling the process noise to the states. For the original tracking algorithm

$$\mathcal{B} = \begin{pmatrix} 0 \\ B \end{pmatrix} \tag{5.56}$$

giving a controllability matrix of

$$\mathcal{C} = \begin{pmatrix} 0 & B \\ B & A_1 B \end{pmatrix} \tag{5.57}$$

which is of full rank when B is. Thus the original tracking algorithm is controllable when the process noise matrix B is invertible.

In the augmented system model, the measurement matrix $\mathcal{B}$ is of the form

$$\mathcal{B} = \begin{pmatrix} 0 \\ B \\ 0 \end{pmatrix}^T, \tag{5.58}$$

giving a controllability matrix of

$$\mathcal{C} = \begin{pmatrix} 0 & B & A_1 B \\ B & A_1 B & A_1^2 B \\ 0 & 0 & 0 \end{pmatrix} \tag{5.59}$$

which is never of full rank. The system is never controllable, which means that the covariance of some state in the state vector will tend to zero. This means that at some point it

would be appropriate to change the dimension of the system being simulated. In practice this turns out to be only five or six timesteps with measurements (if there are no measurements in a given direction, the covariance will never decrease in that direction). Dimension pruning will be discussed in section 5.9.1.

5.7.5 System reformulation to remove singularities in $I - A_0 - A_1$

Much of the discussion about the learning algorithm for the "original" motion model and the augmented motion model has made the assumption that the matrix $\Psi \stackrel{\triangle}{=} I - A_0 - A_1$ is invertible. Experience with learning dynamics on real systems has shown this is often not the case. The consequences of Ψ being singular are many and serious. In the original motion model, $\overline{\mathbf{X}}$ is not recoverable from D. In the augmented motion model, the system becomes unobservable. Additionally, the $\overline{\mathbf{X}}^{\gamma}$ terms cannot be learnt explicitly, so the initial covariance of $\overline{\mathbf{X}}$ can't be computed.

So when is Ψ singular? In the one-dimensional case, $I - A_0 - A_1 = 0$ describes a system where the velocity at time i is equal to the velocity at time $i-1$ times a constant c, or $X_{n+2} - X_{n+1} = c(X_{n+1} - X_n)$. This is similar to an object being driven with a constant force passing through a viscous medium[4]. The important thing is that the forces being applied to the object do not depend on the location of an object. Since the forces are independent of the object's location, the mean of such a system (if it has one) is determined solely by the starting conditions of the system.

This model has four interesting cases: when $c = 1$, the model is constant-velocity. When $c = 0$, the object stops moving instantly, regardless of the previous velocity. If $c \in (0, 1)$, the velocity decreases exponentially, and if $c \in (0, \infty)$ the velocity increases exponentially. The case of $c < 0$ is not interesting, because it is an artifact of the system being discrete and never could occur in a continuous time system. Figure 5.14 shows several such systems.

[4] An object being driven through a viscous medium has a velocity that is changed by an additive constant, so that velocity at time i is the velocity at time $i-1$ plus a constant c. Qualitatively the two have the same motion; quantitatively, the velocity of the $I - A_0 - A_1 = 0$ system changes exponentially

A quick solution to estimating the mean of such a system is to return to the assumption that the mean of the observed sequence is the mean of the system that generated it. However, singularities in Ψ have significant meanings in both the original and augmented motion models, and it is important to examine the properties of such systems and why and how they are degenerate.

Systems with $I - A_0 - A_1 = 0$ (i.e. those degenerate in all directions) share the common property that they have no *a priori* mean. The general form of such systems is

$$\mathbf{X}_{n+2} = (I - A_0)\mathbf{X}_{n+1} + A_0\mathbf{X}_n$$

or

$$\mathbf{X}_{n+2} - \mathbf{X}_{n+1} = -A_0\left(\mathbf{X}_{n+1} - \mathbf{X}_n\right) \tag{5.60}$$

so the behaviour of the system at any given time depends exclusively on the velocity of the system at the previous timestep, and not on the position. It is precisely for this reason that it is impossible to recover $\overline{\mathbf{X}}$ in these systems. They have no *a priori* mean. Consequently, there is no need to use the augmented system to model their mean. Instead, the original motion model can be used which is advantageous since it is a lower dimensional system. A family of some one-dimensional "mean-free" systems is illustrated in figure 5.14.

In the case of multi-dimensional systems, a singular Ψ matrix does not necessarily imply that $\Psi = \mathbf{0}$. For any given system, there may be "mean-free" systems in some directions, but it is quite possible that in other directions there will also be motion that has an *a priori* mean. Empirically, this happens quite frequently, especially when trying to learn dynamics for the augmented motion model. Resorting to the original motion model for the entire system just because one of the system's degrees of freedom is mean-free is overly drastic: it would be more desirable to adapt the augmented motion model to function correctly in the presence of mean-free systems, or better still, derive a hybrid motion model that used the most appropriate motion model for each of the system's degrees of freedom. Both are possible.

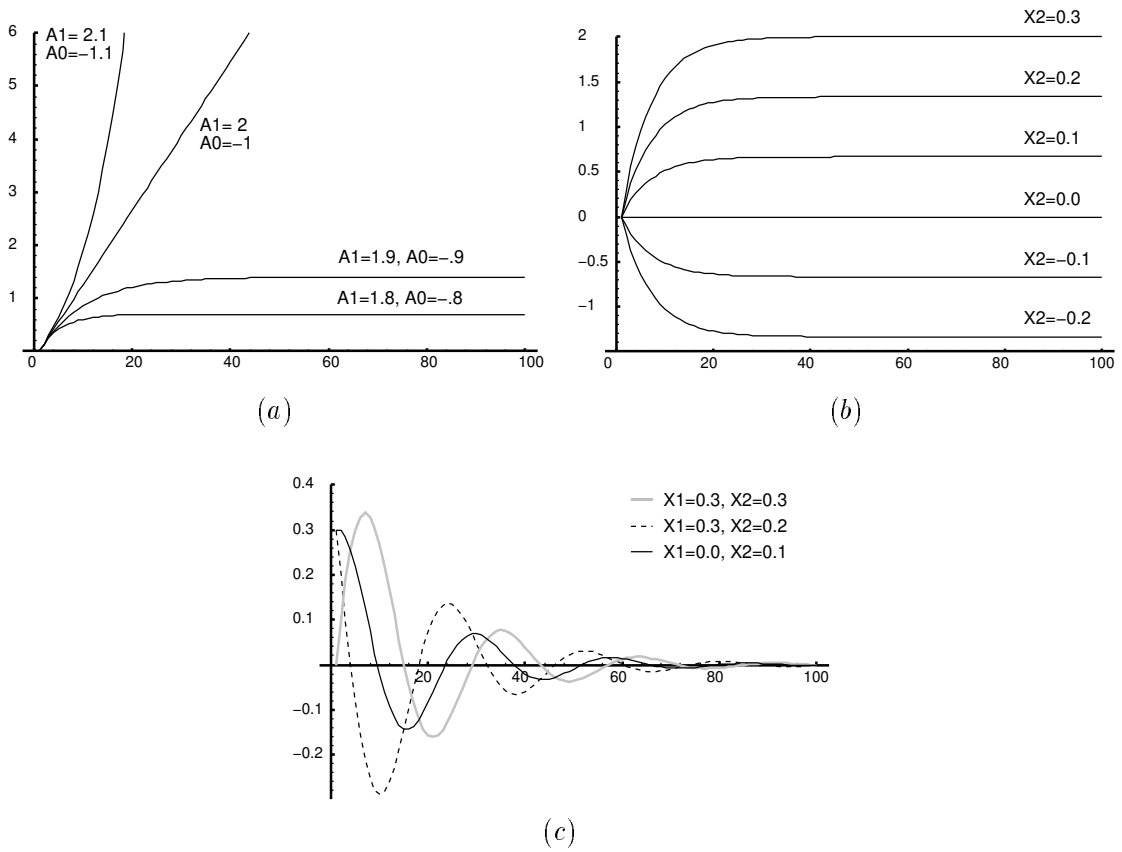


Figure 5.14: **A sample of the family of graphs with $I - A_0 - A_1 = 0$** *Figure 5.14(a) shows the behaviour of some of the dynamics with $I - A_0 - A_1 = 0$. All five traces were generated with the same starting conditions ($X_1 = 0$, $X_2 = 0.14$) and demonstrate the behaviour of the non-stochastic components of the dynamics only. The graph for $A_0 = -1$, $A_1 = 2$ shows constant velocity dynamics. An interesting behaviour of the family is shown in graph 5.14(b). All traces in 5.14(b) have the dynamics $X_{n+2} = 1.85X_{n+1} - .85X_n$ and $X_1 = 0$. Only X_2 is allowed to vary. Unlike most second-order linear differential equations, all equations in this family converge to or diverge from an origin determined by the initial conditions. "Normal" linear second-order systems, such as those illustrated in graph 5.14(c), converge to or diverge from a fixed origin that is independent of the initial conditions.*

One of problems with using the augmented motion model with dynamics that are mean-free is that it is impossible to recover an *a priori* estimate $\overline{\mathbf{X}}$ from the learning. Given Ψ with $\mathsf{Rank}(\Psi) < \mathsf{Dim}(\Psi)$ there exists at least one unit-vector $\mathbf{d}$ such that $\Psi\mathbf{d} = 0$. Without an *a priori* estimate of the value of $\mathbf{d}$, the value of the initial covariance in the direction of $\mathbf{d}$ will be unbounded (i.e. $\mathbf{d}^T P_{init}\mathbf{d} = \infty$). Because the system is not observable in the direction of $\mathbf{d}$, that initial covariance will never decrease.

There are several obvious ways to overcome this. Firstly, it is possible to use the augmented state equations with dynamics learnt from the learning algorithm described in Blake, *et. al.* (1994), which is immune to such singularities. This avoids the issue – it attempts to impose an artificial mean on a system that doesn't have one. The system remains unobservable, so

$$\mathbf{d}^T P_{init} \mathbf{d} = \mathbf{d}^T P_n \mathbf{d} = \infty \quad \forall n. \tag{5.61}$$

A second option is to replace $\overline{\mathbf{X}}$ in the state vector with a different vector of the same dimension that isn't affected by the singularity of $I - A_0 - A_1$. Equations 5.29 and 5.50 show that the variable $\overline{\mathbf{X}}$ never appears by itself in the state equations. Rather, it is always pre-multiplied by Ψ. The new variable $\mathbf{D} \stackrel{\triangle}{=} \Psi\overline{\mathbf{X}}$ (previously defined for an entirely different purpose in equation 5.20) can be substituted in the place of $\overline{\mathbf{X}}$. The system update equation becomes

$$\begin{pmatrix} \mathbf{X}_{n+1} \\ \mathbf{X}_{n+2} \\ \mathbf{D}_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & I & 0 \\ A_0 & A_1 & I \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{X}_{n+1} \\ \mathbf{D}_n \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ B\mathbf{w}_n \\ \mathbf{0} \end{pmatrix}, \tag{5.62}$$

and the measurement process becomes

$$Z = \begin{pmatrix} 0 & I & 0 \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{X}_{n+1} \\ \mathbf{D}_n \end{pmatrix}. \tag{5.63}$$

This system clearly an improvement on equation 5.29: while its controllability is still 0, its observability is $|I| = 1$, so the system is well behaved. The initial value and covariance of $\mathbf{D}$ can be estimated in a similar manner to equations 5.48 and 5.49.

Unfortunately, the new system has more degrees of freedom than the original: figure 5.15 shows an example of a system allowed by this model but not the original model. Any constant offset to the system in the direction of the singularity causes the system to drift over time. In systems that have a mean, a constant offset merely causes a shift in the axis of oscillation. The learning algorithm will assure that the initial estimate of $\mathbf{D}$ and its covariance are set to zero in the appropriate directions. Since the filter has no state noise

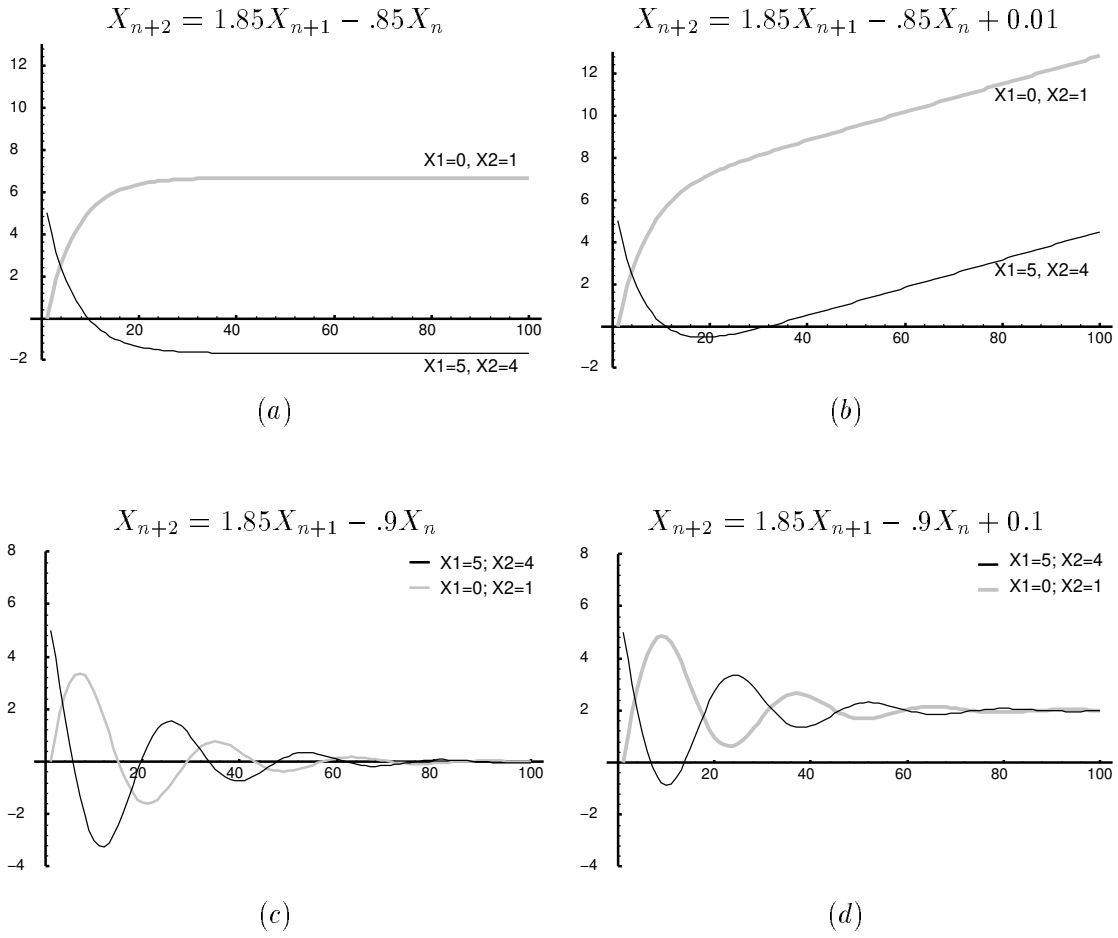


Figure 5.15: **Using D allows illegal systems** *Figure 5.15(a) shows a "mean-free" system under two different starting conditions. When a small constant offset (0.01) is added to the system at each timestep, the system drifts over time (figure 5.15(b)), which is a behaviour not permitted by the original system* $X_{n+2} - \overline{X} = 1.85(X_{n+1} - \overline{X}) - .85(X_n - \overline{X})$. *Figures 5.15(c) and 5.15(d) show the behaviour of two similar systems. Because systems (c) and (d) have* a priori *means, a constant offset in the equation merely changes the axis of oscillation instead of causing the system to drift over time.*

on the **D** term, it will never deviate from its initial estimate. But the Kalman filter still has more degrees of freedom than it needs to model the system.

The removal of this remaining degree of freedom turns out to be quite simple. The matrix Ψ can be written as its singular value decomposition $U^T S V$ where U and V are orthonormal matrices, and S is a diagonal matrix of the singular values. The "significant" components of Ψ can be obtained by truncating S so that it has a reasonable condition number (e.g.

10^{-4}). Then we can define a new variable $\Gamma \triangleq SV\overline{\mathbf{X}}$ with the desired properties:

$$U^T\Gamma = U^T SV\overline{\mathbf{X}} \tag{5.64}$$

$$= \Psi\overline{\mathbf{X}}. \tag{5.65}$$

The system update equations can be rewritten to be

$$\begin{pmatrix} \mathbf{X}_{n+1} \\ \mathbf{X}_{n+2} \\ \Gamma_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & I & 0 \\ A_0 & A_1 & U^T \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{X}_{n+1} \\ \Gamma_n \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ B\mathbf{w}_n \\ \mathbf{0} \end{pmatrix} \tag{5.66}$$

and the measurement process becomes

$$Z = \begin{pmatrix} 0 & I & 0 \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{X}_{n+1} \\ \Gamma_n \end{pmatrix}. \tag{5.67}$$

The observability matrix is

$$\mathcal{O} \triangleq \begin{pmatrix} H \\ HF \\ HF^2 \end{pmatrix} \tag{5.68}$$

$$= \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ A_0 & A_1 & U^T \end{pmatrix} \tag{5.69}$$

so the system is observable if and only if U^T is of full rank. Note that U^T is not square and thus not invertible, but observability depends only on the rank of the observability matrix, not whether it's invertible or not. Now we have a well behaved system that is equivalent to equation 5.62 that is of full rank.

5.8 Results

This chapter has explained in detail the process of learning dynamics from training sequences. Corrections to previous algorithms (Blake and Isard, 1994) have been detailed for multiple training sequences and short training sequences. A new motion model and corresponding learning algorithm has been detailed that provide superior tracking performance by separately modelling class and motion variability.

The augmented system with independently modelled class and motion variability has been applied to the agricultural robotics problem described earlier. The outlines of plants

are described by B-spline curves (Bartels et al., 1987), parameterised here over a six dimensional affine space. The B-spline curves are estimated over time in a Kalman filter tracker in a standard manner (Blake et al., 1993b). In the figures (5.16 and 5.17) below, three different trackers are demonstrated. The first uses a "reasonable" default model, not learnt, but predicting constant velocity and uniform driving noise. The second uses a "joint variability" model in which class and motion variability are lumped together. Because, for this data, class variability swamps motion variability (see figure 5.10), this results in a tracker with constraints on temporal shape change that are unnecessarily weak. Lastly, we demonstrate a tracker based on the new "independent variability" model which exploits the prior knowledge on shape and motion as thoroughly as possible. While the default and joint dynamics trade off performance, the independent variability beats them both.

5.9 Future Work

While the augmented motion model has been proved useful, there are still many questions surrounding its implementation. Because the Kalman filter is an $\mathcal{O}(n^3)$ algorithm, where n is the number of dimensions, it is desirable to reduce n as much as possible while still maintaining tracking performance.

5.9.1 Pruning the dimension of the Kalman filter

The augmented motion model is unusual from a control/filtering standpoint because it is uncontrollable. Each timestep the estimate of the uncertainty of the state ($P_{\mathbf{X}_n\mathbf{X}_n}$, $P_{\mathbf{X}_n\mathbf{X}_{n+1}}$, and $P_{\mathbf{X}_{n+1}\mathbf{X}_{n+1}}$) is increased by the process noise and subsequently decreased by the measurements. Because the estimate of the mean ($\hat{\overline{\mathbf{X}}}$) is uncontrollable, its uncertainty is never increased by the process noise, and because it is observable, measurements reduce its covariance. So eventually the uncertainty $P_{\hat{\overline{\mathbf{X}}}}$ will become zero and the state $\hat{\overline{\mathbf{X}}}$ will effectively become a constant. Empirically, this often occurs within the first second of tracking (see figure 5.13).

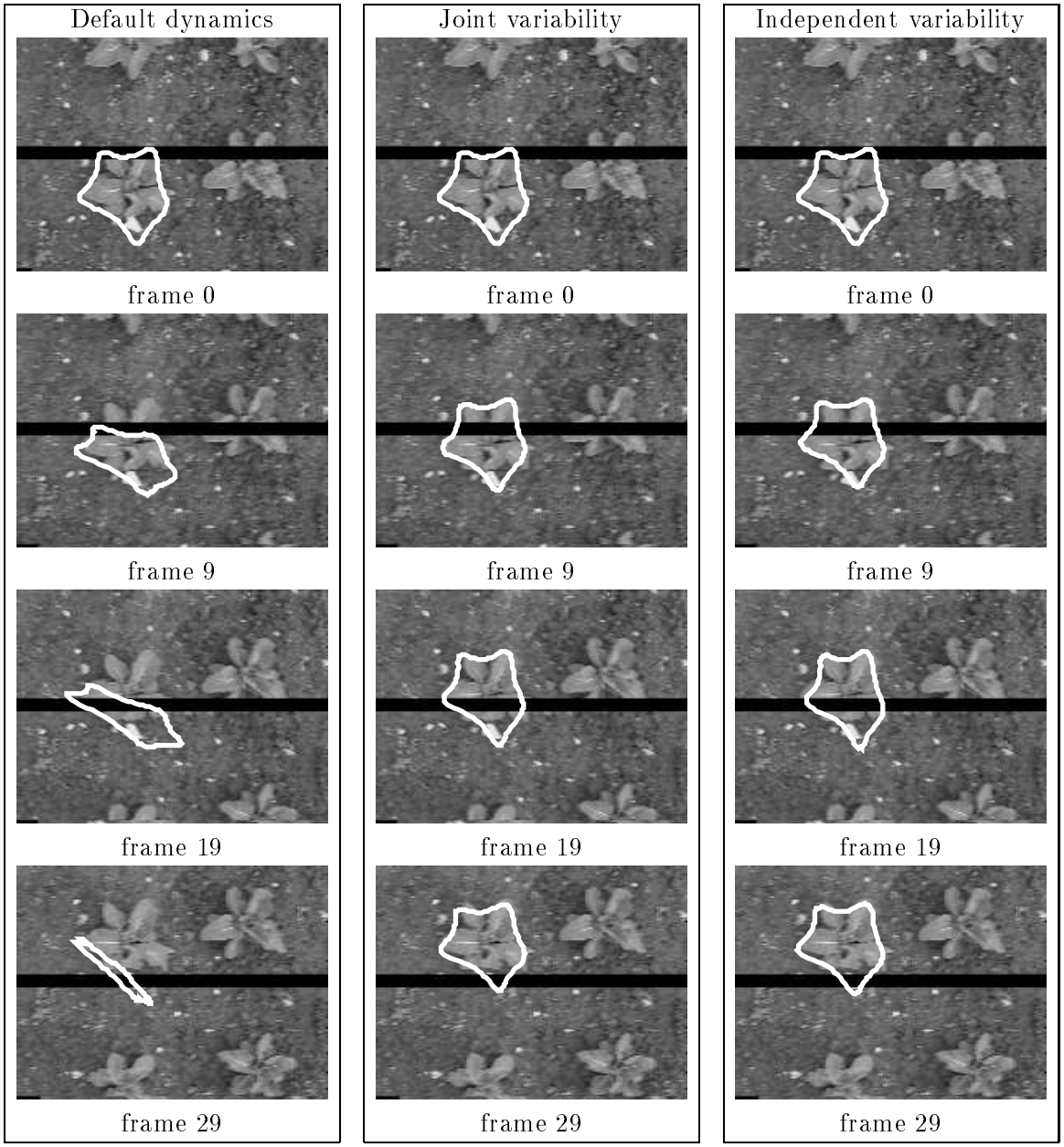


Figure 5.16: **Stability from learnt dynamics.** *An image sequence is tracked over time (top to bottom) with three different trackers. A black bar has been superimposed on the data to add (in exaggerated form) the effect of disturbances such as shadows. First a tracker with default, constant velocity, dynamics allows too much shape freedom and lock is lost. Secondly and thirdly, tracking with learnt dynamics is successful because constraints on shape change have been learnt from a training set. In this data set, the size of the plant is about average, so the "joint variability" model succeeds despite the weakness of its shape constraints.*

Once the mean has converged to a fixed value, it will be desirable to automatically reconfigure the filter so that it stops estimating $\hat{\overline{\mathbf{X}}}$. Because the augmented motion model increases the dimension of the state by one-half, switching the filter back to the normal

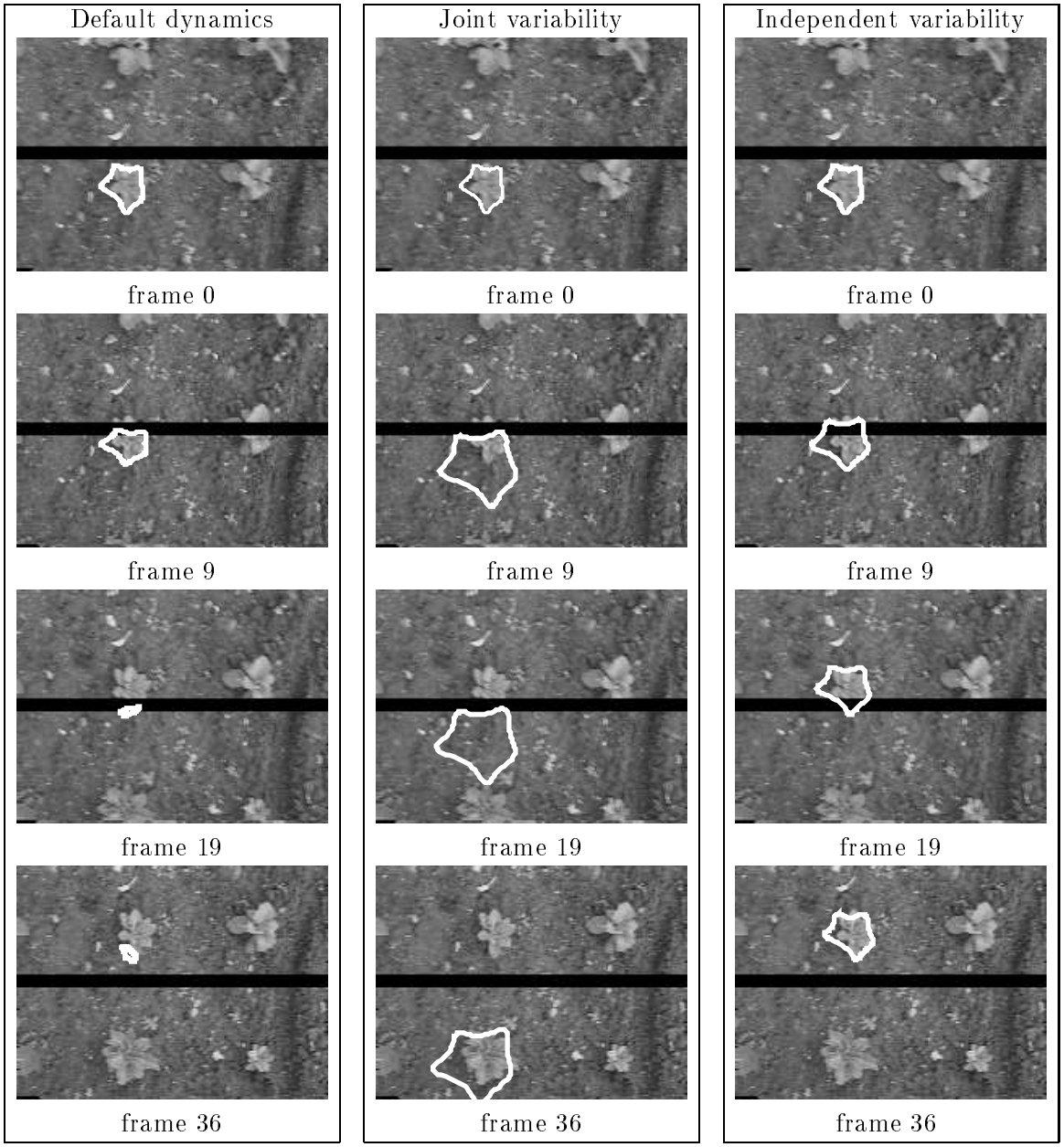


Figure 5.17: **Joint variability modelling is too weak.** *Results are shown here for the three trackers, as in figure 5.16 but for a different test sequence. In this case the tracked plant is rather smaller than average. The "joint variability" tracker reverts, after a short time, towards the grand mean of the training set and hence loses lock on the plant. The "independent variability" tracker however adjusts rapidly, via its additional mean-configuration variable* $\overline{\mathbf{X}}$*, to the small size of this plant and continues to track successfully.*

motion model should make the Kalman filter run $\left(\frac{3}{2}\right)^3 \approx 3$ times as fast. The research question is how to decide when to switch the model.

One possible solution would be to switch the filter after a fixed number of timesteps.

Of course the number of timesteps would have to depend on the particular dynamics being used – the mean process of figure 5.12 converges much slower than the mean process of figure 5.13. Unfortunately, the presence or absence of measurements will also greatly affect the rate the uncertainty decreases. This implies that switching after a fixed number of timesteps may not be the best idea.

A second possibility would be to examine the uncertainty of $\overline{\hat{\mathbf{X}}}$ and switch the filter based on the magnitude of the largest eigenvalue or the condition number. Some sort of metric would have to be used to make sure that dimensions sensitive to small changes (such as magnification number) weren't compared directly with dimensions relatively insensitive to small changes (such as translation in pixels). A suitable metric may be the W matrix that specifies the relationship between the control points and the state space of the filter. By examining the uncertainty of $\overline{\hat{\mathbf{X}}}$ in the control point space, it's possible to set a more meaningful cutoff. For example, the filter could be switched when the standard deviation of a dimension is less than one pixel on all the control points.

Ideally, such a scheme would also take into account how much the current state uncertainty is affecting the tracking performance. If a particular process has a large process noise or has dynamics that are near $I - A_0 - A_1 = 0$, it will be relatively insensitive to small changes in $\overline{\hat{\mathbf{X}}}$.

This brings up the largest unanswered question. While the augmented motion model has been proved to be useful in certain circumstances, it almost certainly will not be useful in other circumstances. In the same way that an analysis of the process noise, state uncertainty and dynamics could produce a criterion for switching from the augmented motion model to the normal motion model, an analysis should be able provide quantitative evidence for whether the augmented motion model is appropriate at all for a given training set.

In figure 5.13, $\overline{\hat{\mathbf{X}}}$ seems to be supplying nearly as much information as $\hat{\mathbf{X}}$. A probabilistic analysis should also be able to say whether $\hat{\mathbf{X}}$ is of the right dimension or not. It is reasonable to assume that it may be useful to estimate $\overline{\hat{\mathbf{X}}}$ without estimating $\hat{\mathbf{X}}$ – the

current formulation of the augmented motion model doesn't allow this, though a slight correction to the H matrix should be sufficient.

All these problems seem to be essentially a principal components analysis in the state space of the filter. It is not clear exactly which norms should be used or what the criteria for inclusion or exclusion of a particular principal component are. But the computational expense of a Kalman filter makes this sort of analysis important.

6

The Initialisation Problem and Active Contours

The Active Contour paradigm of visual tracking has shown itself to be a reliable method for tracking complicated shapes and motion using a minimal amount of computational resources. Once an active contour has locked on to its target, it can follow the target through diverse motions and levels of clutter. In the past, the initialisation problem, that of getting the active contour locked onto its target in the first place, has largely been ignored. Either the operator had to start the tracking system when the target passed directly under the active contour, or the target was required to enter the field of view of the camera at specific locations.

The computational engine behind the Active Contour is the Kalman filter. The initialisation of the Active Contour can be broken down into two distinct processes: that of initialising the position of the contour (the state estimate of the Kalman filter) and that of initialising the uncertainty about the position (the state covariance of the Kalman filter).

The first process involves finding the target in the first place with no or little *a priori* knowledge. Of course this is much more difficult than finding the target in subsequent

frames. Known as *pose recovery*, the problem is a fundamental one in computer vision. In the case of an Active Contour, this problem is more difficult because of the sparse representation of the shape being tracked and the strong constraints on the ways that shape is allowed to deform. Only the occluding contour of the object is known ahead of time, so methods of pose recovery that rely on strong models of the appearance of the object will be of no help. The methods of roughly localising the target are discussed in section 6.2.

Once the target is localised, finding the B-spline deformation that best fits the target is another difficult problem. This will be discussed in section 6.1. Fortunately, the target is only required to be located approximately, which means that in many cases, simple pose recovery techniques are suitable. These will also be discussed in section 6.2.

Estimating the initial uncertainty of the Kalman filter is in many ways more difficult than estimating the initial state, because it requires the uncertainty of the original estimate to be quantified. While estimating the uncertainty may seem to be of little importance, an accurate estimate of the initial covariance can be critical to the performance of the Kalman filter. Overestimating uncertainty (i.e. assuming the state estimate is worse than it is) can allow the filter to lock on to false features, while underestimating uncertainty (i.e. assuming the state estimate is better than it is) causes the filter to ignore the images it is trying to track. Techniques for estimating the uncertainty will be discussed in chapter 7.

This chapter begins with a discussion of some difficulties involved with fitting B-splines to arbitrary curves in images. A series of methods is presented for fitting B-splines to curves by measuring the moments of area of the interior of the curve. When the B-splines are constrained to lie in certain special spaces, such as the space of affine deformations of a template, the solution of the problem is analytic. The general case of fitting a B-spline constrained to a non-affine space is shown to be analytically insoluble. However, several approximations to the solution are presented using both analytic and numerical methods. Results of the initialisation algorithms are shown for a variety of problem types. The chapter concludes with a discussion of estimating the uncertainty of the initial state.

Several methods, both common and original, are presented, as well as a novel method for simultaneously initialising the state and uncertainty of the system.

6.1 B-splines and parameter dependences

Active Contours use Kalman filters to estimate the position of a B-spline in a sequence of images. The state at each time step corresponds to the deformation of the B-spline that best approximates the occluding contour of the target. Thus, estimating the initial position of the B-spline is a problem of finding the target in the image and finding the B-spline that best approximates it.

One fundamental problem with fitting a B-spline to a curve is the parameter dependence of the B-spline. The closed, quadratic B-spline is a parametric function defined as

$$\mathbf{Q}(t)^T \triangleq \begin{pmatrix} 1 & (t-\lfloor t \rfloor) & (t-\lfloor t \rfloor)^2 \end{pmatrix} B \begin{pmatrix} \mathbf{q}_{\lfloor t \rfloor}^T \\ \mathbf{q}_{(\lfloor t \rfloor+1) \bmod N}^T \\ \mathbf{q}_{(\lfloor t \rfloor+2) \bmod N}^T \end{pmatrix}, \tag{6.1}$$

where $\mathbf{q}_i$ are the control points of the B-spline, and the matrix B depends on the order of the knots of the control points. For simplicity, in this chapter all B-splines will always be assumed to be closed, quadratic and have no multiple knots. In this case, the B-spline equation becomes

$$\mathbf{Q}(t)^T = \begin{pmatrix} 1 & (t-\lfloor t \rfloor) & (t-\lfloor t \rfloor)^2 \end{pmatrix} \begin{pmatrix} 0.5 & 0.5 & 0 \\ -1 & 1 & 0 \\ 0.5 & -1 & 0.5 \end{pmatrix} \begin{pmatrix} \mathbf{q}_{\lfloor t \rfloor}^T \\ \mathbf{q}_{(\lfloor t \rfloor+1) \bmod N}^T \\ \mathbf{q}_{(\lfloor t \rfloor+2) \bmod N}^T \end{pmatrix}. \tag{6.2}$$

While it is easy to find the curve described by a B-spline, the inverse process (that of finding a B-spline that best describes a curve) is not. One of the first problems is that there are many different B-splines representations for each contour. This is illustrated in figure 6.1. The multiple representations mean that it is difficult to define even simple operations, such as a distance measure.

6.1.1 A parameterisation invariant distance function

The first step towards approximating curves with B-splines is to define an appropriate distance function $D(\mathbf{r}, \mathbf{q})$ to assess the goodness of fit of a B-spline to a curve. In this case, $\mathbf{r}$ is

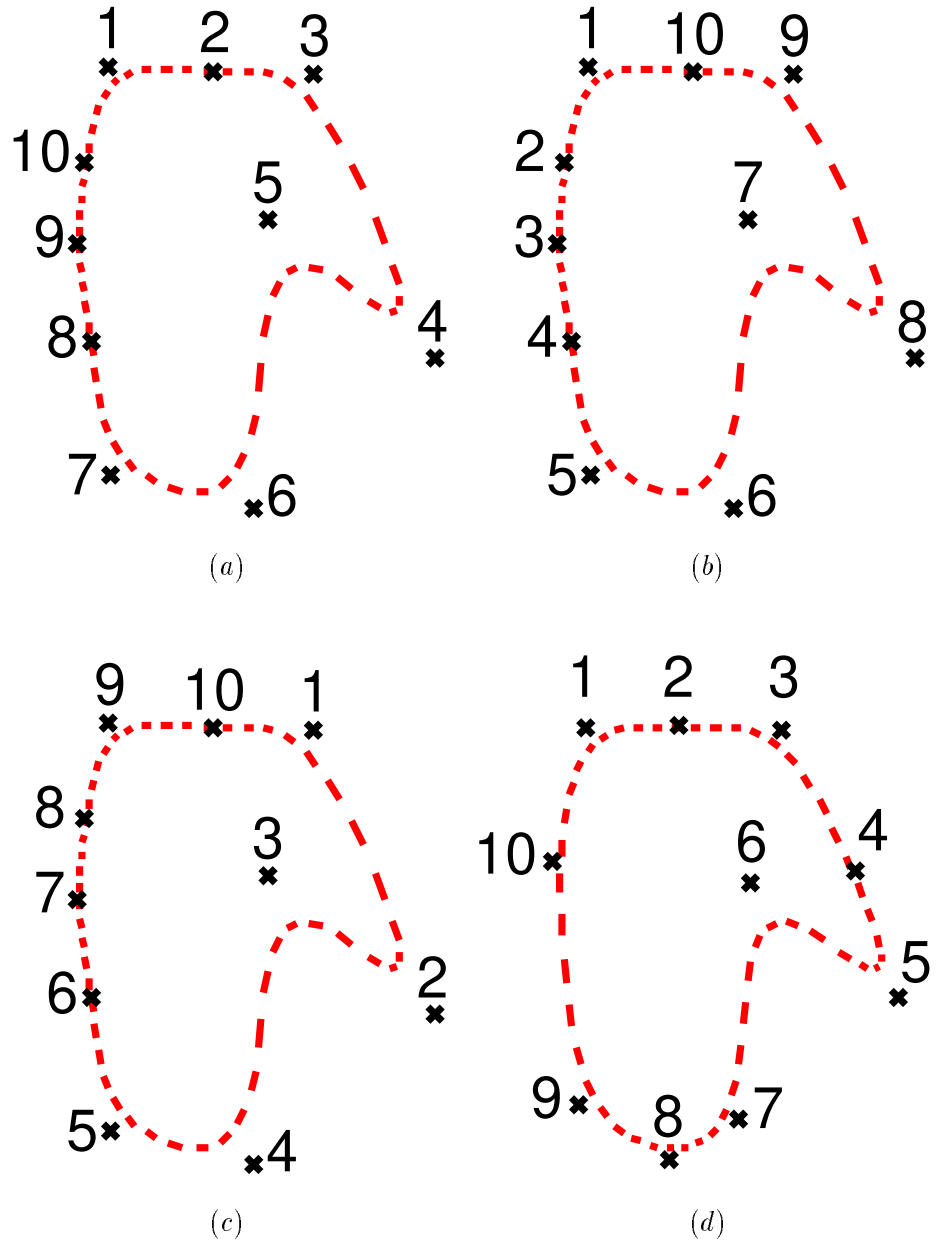


Figure 6.1: **A given contour has many B-spline representations** *One of the properties of B-splines is that it is easy to obtain a contour from a B-spline, but difficult to obtain a B-spline from a contour. This, in part, is because there are many different B-spline representations of every contour. Reversing (figure 6.1b) and cycling (figure 6.1c) the control points (shown as crosses) are two such operations that do not affect the contour generated by the B-spline. However, there are other more complicated reparametrisations that give contours that are very similar (figure 6.1d).*

a curve with an arc-length parametrisation, and $\mathbf{q}$ is one with an arbitrary parametrisation. It is important that this distance function have certain characteristics: (1) It should be invariant to the way the curve $\mathbf{q}$ is parametrised. For example, the distance between all the B-splines in figure 6.1 should be small. (2) The distance should be influenced by all the points on both curves, so that no points are ignored or discarded. (3) The function should be able to compare curves that are 'close' to each other, and there should be a well-defined criterion that states when the curves are close enough. (4) D should be non-negative and be zero only when the two curves are identical. The resulting function need not be a metric because $D(\mathbf{r}, \mathbf{q})$ does not necessarily equal $D(\mathbf{q}, \mathbf{r})$.

Given two parameterised closed curves $\mathbf{r}(s)$ and $\mathbf{q}(t)$ where $s \in [0, \sigma)$ and $t \in [0, \tau)$, the goal is to find a function D that satisfies the above criteria. For simplicity, $\mathbf{r}(s)$ and $\mathbf{q}(t)$ are assumed to be 1-1, so that

$$\mathbf{r}(s_1) \neq \mathbf{r}(s_2) \quad \forall s_1 \neq s_2$$

$$\mathbf{q}(t_1) \neq \mathbf{q}(t_2) \quad \forall t_1 \neq t_2$$

To find such a D, a reparametrisation $p(s)$ is needed that takes any point on $\mathbf{r}$ and maps it to its 'equivalent' point on $\mathbf{q}$. These two points effectively become paired in the distance function. The reparametrisation can then be used to integrate the distance between each point on $\mathbf{r}$ and its pair on $\mathbf{q}$.

First, D is defined to be a non-symmetric function of $\mathbf{r}$ and $\mathbf{q}$:

$$D(\mathbf{r}, \mathbf{q}) = \min_{p} \int_{o}^{\sigma} \| \mathbf{r}(s) - \mathbf{q}(p(s)) \|^2 \, ds. \tag{6.3}$$

It is clear that all functions of this kind satisfy criterion (4). A few constraints on p are introduced to ensure all points on both curves have pairs (and meet criterion (2)):

$$\begin{aligned} p(0) &= \gamma; \\ p(\sigma) &= \gamma + \tau; \\ p(s) & \quad \text{is continuous and monotonic.} \end{aligned}$$

This function D is then invariant to the parametrisation of the two curves and can be used as the distance function. Consider another parametrisation $\mathbf{q}^*(t^*)$ $t^* \in [0, \tau^*]$ of the curve $\mathbf{q}$ and the function $p^*(t)$ that relates points on $\mathbf{q}$ to $\mathbf{q}^*$. It is clear that if p minimises the distance $D(\mathbf{r}(s), \mathbf{q}(p(s)))$ then $p^*(p)$ will minimise the distance $D(\mathbf{r}(s), \mathbf{q}^*(p^*(p(s))))$ and that those two distances will be the same. This proves the invariance of D to the parametrisations of $\mathbf{q}$.

This function D is a function of the form $J = \int_{x_1}^{x_2} f(y, x)dx$, where

$$f \stackrel{\triangle}{=} \| \mathbf{r}(s) - \mathbf{q}(p(s)) \|^2 \tag{6.4}$$

and $x_1 = 0$ and $x_2 = \sigma$. By the calculus of variations, this expression is minimised when

$$\frac{\partial f}{\partial y} = 0. \tag{6.5}$$

This leaves the constraint

$$\begin{aligned} 0 &= \frac{\partial}{\partial p} \left(\| \mathbf{r}(s) - \mathbf{q}(p(s)) \|^2 \right) \\ &= 2 \left[\mathbf{r}(s) - \mathbf{q}(p(s)) \right] \cdot \frac{\partial}{\partial p} \left[\mathbf{r}(s) - \mathbf{q}(p(s)) \right] \\ &= -2 \left[\mathbf{r}(s) - \mathbf{q}(p(s)) \right] \cdot \mathbf{q}'(p(s)), \end{aligned} \tag{6.6}$$

where $\mathbf{q}'(p(s)) = \partial \mathbf{q}(p(s))/\partial p(s)$ which is along the tangent of $\mathbf{q}$ at the point $p(s)$. If there is a function $p(s)$ that minimises D in equation 6.3, then it must satisfy this constraint for all s. Thus, if a parameterisation p is one of minimum distance, p must be such that every point $\mathbf{q}(t)$ must correspond to a point on $\mathbf{r}$ such that the vector connecting them is normal to $\mathbf{q}$ at t. Thus any good minimum distance function must use measurements in the normal direction.

Now it is possible to say exactly when this minimum distance function fulfills criterion (2): Firstly, if there is any point $\mathbf{q}(p(s))$ such that the line normal to $\mathbf{q}$ at $\mathbf{q}(p(s))$ does not intersect $\mathbf{r}$, there is no reparametrisation that provides a minimum distance function. In this case, p is not a function because it is not single-valued – it tries to map individual

points of **r** onto a range of points on **q**. This is illustrated in figure 6.2a. Secondly, if p does not have an inverse, the parametrisation function is not one-to-one. This means that multiple points on the curve **q** are being paired with the same point on **r**. While this solution still satisfies all the assumptions made by the the calculus of variations, it is not useful for a distance function because it violates criterion (2), so it can be disallowed as a parametrisation (see figure 6.2(b)).

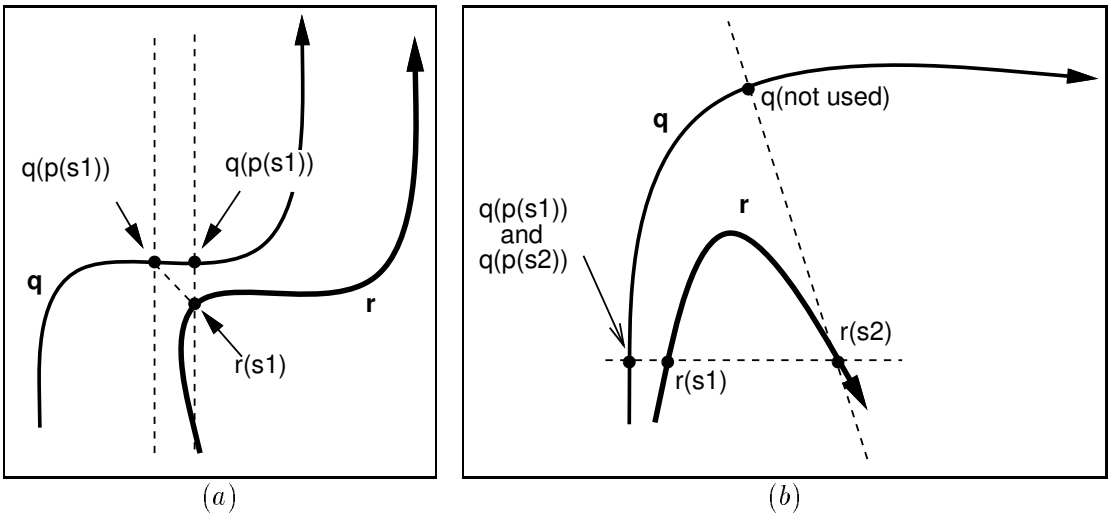


Figure 6.2: **Degeneracies of the parametric distance function** *Given two curves* **r** *and* **q**, *it is possible to use the calculus of variations to derive the reparametrisation function* $p(s)$ *such that* $\int_o^\sigma \parallel \mathbf{r}(s) - \mathbf{q}(p(s)) \parallel^2 ds$ *is minimised. The derivation indicates there are two times when there is no function* $p(s)$ *that minimises the distance. If there is a point on* **q** *such that the line normal to it does not intersect* **r**, *the reparametrisation function* p *is not a function because it maps multiple points of* **q** *onto the same point of* **r**. *Thus* $p(s)$ *is not a function at all (figure 6.2(a)). Similarly it is possible for* p *to map multiple points on the curve* **r** *onto the same point on* **q**. *In this case* p *is not 1-to-1 or onto, and thus doesn't have an inverse (figure 6.2(b)). See section 6.1.1 for more details.*

6.1.2 Fitting B-Splines to arbitrary curves

In the same way that measuring the distance between two B-splines is difficult, finding the best B-spline approximation to a generalised curve or a set of discrete points is a difficult problem (Diercks, 1993; Duda and Hart, 1973; Cham and Cipolla, 1996). Many methods use *ad hoc* and iterative approximate solutions, since the 'optimal' solutions are difficult if not impossible to find. The problem of finding the initial position of the contour in the

Active Contour model is even more difficult, since the fit B-spline is constrained to lie in a pre-defined shape space that is much smaller than the space of all B-splines of a particular order.

6.2 Initialisation of the Kalman filter

The state of the Kalman filter can be broken up into two basic parts: the state estimate $\hat{\mathbf{X}}$ and the state covariance P. Before using the filter, both parts must be given initial values:

$$\hat{\mathbf{X}}_{initial} = \mathsf{E}(\mathbf{X}_0); \tag{6.7}$$

$$P_{initial} = \mathsf{E}\left((\mathbf{X}_0 - \hat{\mathbf{X}}_{initial})^2\right). \tag{6.8}$$

The simplicity of the equations obscures the complexity of the problem; the process of obtaining good values for these two variables is actually quite difficult. As a result, arbitrary values are often assigned to these two variables in the hopes that the Kalman filter will quickly recover from this initial error. This is true even in the 'applications' sections of some textbooks about Kalman filters (Grewal and Andrews, 1993). It will be shown that in the case of Active Contours, a large initial error in the state estimate and state uncertainty can cause tracking to fail.

There are three basic assumptions that can be made about the initial state of the Kalman filter. The first assumes that the position is known *a priori*. In this case, the initial state estimate is a constant, and the filter can start running immediately. It is discussed in section 6.2.1. The second assumption is that at some point in the future, the state will be a value that is known *a priori*. The initial state of the Kalman filter is known *a priori*, but the filter cannot start running until the system's state corresponds to that of the filter. This is discussed in section 6.2.2. The third assumption is that the initial state is unknown and must be estimated online through some sort of pose-recovery algorithm. This is discussed in section 6.2.3.

6.2.1 No initialisation

By assuming that the state of the Kalman filter is known at the moment that the filter starts running, the whole problem of estimating the initial state is eliminated. In many cases this is not an unreasonable assumption to make and many successful tracking systems have made it (Blake et al., 1993a; Baumberg and Hogg, 1995b; Cootes and Taylor, 1992; Ahmad et al., 1995). Sometimes this is achieved by operator intervention: a human is required to manually initialise the system. Lanitis, *et. al.* (Lanitis et al., 1995) use active shape models to track and interpret facial images. For initialisation, their tracker is allowed to iterate until it converges on the desired features. Isard and Blake (Isard and Blake, 1996) use the Condensation tracker to obtain 'blue-screen' effects from a pre-recorded image sequence. In such a case, when the tracking sequence is pre-recorded, this sort of initialisation may be completely acceptable.

If the starting estimate is not good enough, the Active Contour can fail. The contour can lock onto the wrong edge of the target or start tracking a local minimum that is different than the global minimum. The contour can also lock onto background features instead of the target. Contours are unlikely to recover from either of these failures.

6.2.2 Motion alarms

While the previous section described a way to initialise the Active Contour when the position is known at particular time, a slightly more general system results from assuming that the position of an object is known *a priori* at some undetermined time in the future. Ferrier and Rowe (Ferrier et al., 1994) found that in the application of car tracking, they could obtain reliable automatic initialisation of the active contour by placing a static car-shaped template at a place where they knew cars would be entering the image. They computed image differences underneath the template, and when the image difference was above a certain threshold it was assumed that a car had passed underneath the template. This 'motion alarm' started the tracking mechanism.

6.2.3 Automatic recovery of the starting state

In the case where there is little or no *a priori* information about the initial state of the Kalman filter, it is necessary to obtain an initial state estimate from some other method. This method must (1) detect the presence of a potential target, (2) check that the potential target is a reasonable one, (3) recover the state estimate corresponding to the location of the target (4) recover the uncertainty of that estimate. The following sections primarily describe efforts to resolve (3), whereas (4) is treated in chapter 7.

6.2.4 Moment based initialisation

One strategy to solve the initialisation problem involves using the moments of inertia as a metric for recovering transformations. The use of moments in computer vision for pose recovery and pattern recognition in computer vision is wide spread and well documented (Abu-Mostafa and Psaltis, 1985; Teh and Chin, 1988; Blake and Marinos, 1989; Modayur et al., 1992; Prokop and Reeves, 1992; Sato and Cipolla, 1994; Abdallah et al., 1994). The geometric, or normal moments of order $(p+q)$ of an image $f(x,y)$ are defined as

$$M_{pq} \triangleq \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)\,dx\,dy \quad p,q \in \mathbb{Z}. \tag{6.9}$$

The basis set of geometric moments $\{x^p y^q\}$ is known to be complete but not orthogonal. There are many other kinds of moments that use different sets of basis vectors to attain different properties. The different kinds of moments and their properties are discussed at length in (Teh and Chin, 1988) and (Prokop and Reeves, 1992). Only geometric moments will be discussed in this chapter, and hereafter 'moments' refers to geometric moments.

Given a model of an object (a template) it is possible to measure that object's moments. Similarly, it is possible to measure the moments of that object after it has undergone a known transformation. For certain classes of transformations, it is possible to recover the transformation just from comparing the moments of the object before and after it has been transformed. The class of similarity transformations (translation, rotation and scaling) and

affine transformations (similarity transformations plus shearing) are two such classes. For other classes of transformations it is possible to recover part but not all of the transformation by comparing first and second-order moments.

Active Contours make strong assumptions about the shapes a target is allowed to attain during a tracking sequence. These assumptions are specified by defining a template as the 'base' shape of an object, and a family of deformations this template is allowed to undergo. In the case that an image can be segmented using some other method (such as the pixel-based classification discussed in chapter 3) into a 'foreground' (target) and 'background' (non-target) it is possible to compute the moments of the target in the image (i.e. the moments of the deformed template) and recover some or all of the deformation relating the template to the image. This recovered deformation can then be used to initialise the state estimate of the Kalman filter.

This section discusses the case when the family of transformations relating the template to an image is the family of affine transformations. Section 6.3 will discuss methods to solve the problem in the case of a more general set of transformations.

The family of affine transformations is a very useful set of transformations because they correspond to the deformations a 2D object can undergo whilst being imaged by a camera when that camera is far away compared to the depth of the object (i.e. the depth of the object is small compared to the viewing distance) (Mundy and Zisserman, 1992; Faugeras, 1993). This camera model is known as the affine camera or 'weak perspective' model. The affine transformations have been found to be a useful basis for the problems of computing image flow (Koenderink, 1986), object recognition (Hopcroft et al., 1992), and tracking (Blake et al., 1993a).

With an affine transformation, coordinates in the transformed shape $\begin{pmatrix} x' & y' \end{pmatrix}^T$ are related to coordinates in the original (template) shape $\begin{pmatrix} x & y \end{pmatrix}^T$ by the following relation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \triangleq \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix} + \mathbf{t}. \tag{6.10}$$

For the purposes of recovering affine transformations, it is simpler to compare moments that have been normalised by the zeroth M_{00} and first M_{10}, M_{01}:

$$c_0 \triangleq \iint_R 1 \, dx \, dy \tag{6.11}$$

$$\mathbf{c} \triangleq \begin{pmatrix} c_x \\ c_y \end{pmatrix} = \frac{1}{c_0} \iint_R \mathbf{x} \, dx \, dy \tag{6.12}$$

$$C \triangleq \frac{1}{c_0} \begin{pmatrix} c_{xx} & c_{xy} \\ c_{yx} & c_{yy} \end{pmatrix} = \frac{1}{c_0} \iint_R (\mathbf{x} - \mathbf{c})(\mathbf{x} - \mathbf{c})^T \, dx \, dy. \tag{6.13}$$

These normalised moments have familiar definitions: c_0 is the area of the target, $\mathbf{c}$ is its centroid, and C is the target's moments of inertia matrix.

The way that an affine transformation affects the normalised moments is quite straightforward. For simplicity, assume that the template has a centroid of zero (i.e. $\mathbf{c} = \mathbf{0}$, or the object is centred on the origin). This can be done without loss of generality because it is always possible to subtract the centroid from the template, and it has the advantage of greatly simplifying the mathematics. Now the moments of the image (written as $c_0', \mathbf{c}'$ and C') can be written as a function of the moments of the template and the parameters of the

affine deformation:

$$
\begin{aligned}
c_0' &= \iint_R 1\, dx'\, dy' = \iint_R 1 \det(A)\, dx\, dy = \det(A) c_0 && (6.14)\\
\mathbf{c}' &= \frac{1}{c_0'} \iint \mathbf{x}'\, dx'\, dy' \\
&= \frac{1}{\det(A) c_0} \iint (A\mathbf{x} + \mathbf{t}) \det(A)\, dx\, dy \\
&= \frac{\det(A)}{\det(A) c_0} \left(\iint A\mathbf{x}\, dx\, dy + \iint \mathbf{t}\, dx\, dy \right) \\
&= A \frac{1}{c_0} \iint \mathbf{x}\, dx\, dy + \frac{c_0}{c_0} \mathbf{t} \\
&= A\mathbf{c} + \mathbf{t} = \mathbf{t} && (6.15)\\
C' &= \frac{1}{c_0'} \iint_R (\mathbf{x}' - \mathbf{c}')(\mathbf{x}' - \mathbf{c}')^T\, dx'\, dy' \\
&= \frac{1}{\det(A) c_0} \iint_R (A\mathbf{x} + \mathbf{t} - \mathbf{c}')(A\mathbf{x} + \mathbf{t} - \mathbf{c}')^T\, dx'\, dy' \\
&= \frac{1}{\det(A) c_0} \iint_R (A\mathbf{x} + \mathbf{t} - \mathbf{t})(A\mathbf{x} + \mathbf{t} - \mathbf{t})^T\, dx'\, dy' \\
&= \frac{1}{\det(A) c_0} \iint_R (A\mathbf{x})(A\mathbf{x})^T \det(A)\, dx\, dy \\
&= \frac{\det(A)}{\det(A) c_0} A \left(\iint_R \mathbf{x}\mathbf{x}^T\, dx\, dy \right) A^T \\
&= ACA^T && (6.16)
\end{aligned}
$$

There are several things to notice about the above equations. First of all, c_0' and C' are not affected by translation $\mathbf{t}$. So by solving equation 6.15 first, it is possible to recover $\mathbf{t}$, and then use equation 6.16 to recover A.

One way to recover A, is to find the square roots[1] of the matrices C and C'. Since C and C' are both symmetric 2×2 matrices, there is a closed form for a square root of these

[1] This is a slightly non-standard usage of the term "matrix square root". For this discussion, B is a square root of A if $BB^T = A$. Some standard references call C a square root of A if $CC = A$, and either don't give $BB^T = A$ a name (Strang, 1976; Barnett, 1990), or call it a Cholesky Factor (Grewal and Andrews, 1993). Others (Press et al., 1988) call the Cholesky Decomposition ($LL^T = A$ where L is lower triangular) the "matrix square root".

matrices:

$$P \triangleq \frac{1}{\sqrt{c'_{xx}}} \begin{pmatrix} c'_{xx} & 0 \\ c'_{xy} & \sqrt{c'_{xx}c'_{yy} - (c'_{xy})^2} \end{pmatrix} \tag{6.17}$$

$$Q \triangleq \frac{1}{\sqrt{c_{xx}}} \begin{pmatrix} c_{xx} & 0 \\ c_{xy} & \sqrt{c_{xx}c_{yy} - (c_{xy})^2} \end{pmatrix} \tag{6.18}$$

Since $C' = PP^T$ and $C = QQ^T$, P is a square root of C' and Q is a square root of C. Because C' and C have only 3 degrees of freedom, and A has 4 degrees of freedom, there is a one degree of freedom family of solutions for A. If $C = QQ^T$ then $C = QOO^TQ^T$ for any orthogonal matrix O, so QO is also a square root of C. Therefore

$$\begin{aligned} C' &= AQOO^TQ^TA^T \\ PP^T &= AQOO^TQ^TA^T \end{aligned} \tag{6.19}$$

for any orthogonal O. It can be seen that

$$A = PO^{-1}Q^{-1} = PO^TQ^{-1}, \tag{6.20}$$

is a solution to equation 6.19, where O is now fixed but unknown.

Of course now there remain some ambiguities. In equations 6.17 and 6.18, it is possible to choose the positive or negative value of the square roots $\sqrt{c_{xx}}$ and $\sqrt{c'_{xx}}$. However, these degrees of freedom are subsumed by O, since multiplying the entire matrix by -1 is the same as rotating the orthogonal matrix by π radians. The square root in the bottom right corner of the matrices corresponds to a reflection about the x-axis.

It might be hoped that equation 6.14 could be used to recover O. Unfortunately equation 6.14 only tells us the value of $\det(A)$ which has already been determined by equation 6.20:

$$\det(A) = \det(PO^TQ^{-1}) = \det(P)\det(O^T)\det(Q^{-1}) \tag{6.21}$$

Since $\det(P)$ and $\det(Q^{-1})$ are fixed, and $\det(O^T) = 1$, equation 6.14 is redundant. To completely recover an affine deformation, it is necessary to either use moments higher than second order or some other form of information about the system.

6.2.5 The canonical frame

The family of solutions corresponding to the "missing" degree of freedom can be easily visualised once the equations are examined more closely. The system as specified by the first and second order moments approximates the shape of the area by an ellipse in two dimensional space. The missing degree of freedom corresponds to an unknown rotation within that ellipse.

Since this 'rotation within an ellipse' is relatively complicated to visualise, it is helpful to transform the template and image to an intermediate basis, a canonical frame. For all ellipses, there exists an affine transformation that transforms the ellipse into the unit circle. By a similar argument, there exists an affine transformation that transforms the image into a frame where $c'_{xx} = 1$, $c'_{yy} = 1$ and $c'_{xy} = 0$. Since $A = PO^TQ^{-1}$, multiplying by Q will accomplish this. A different affine transformation will transform the template into a frame where $c_{xx} = 1$, $c_{yy} = 1$ and $c_{xy} = 0$. Since $A^{-1} = QOP^{-1}$, multiplying by P will accomplish this.

In this canonical frame, the image and template differ only by a rotation. The family of solutions from equation 6.20 corresponds to these rotations, and is parameterised by θ. Since the relationship between the image and template is so simple, the canonical frame is useful both for visualising the relationship between the two, as well as for solving for the missing degree of freedom.

Figure 6.3 illustrates the transformation to the canonical frame of an image and its template. It also shows some of the family of solutions that remain after matching five of the six affine degrees of freedom using equation 6.20.

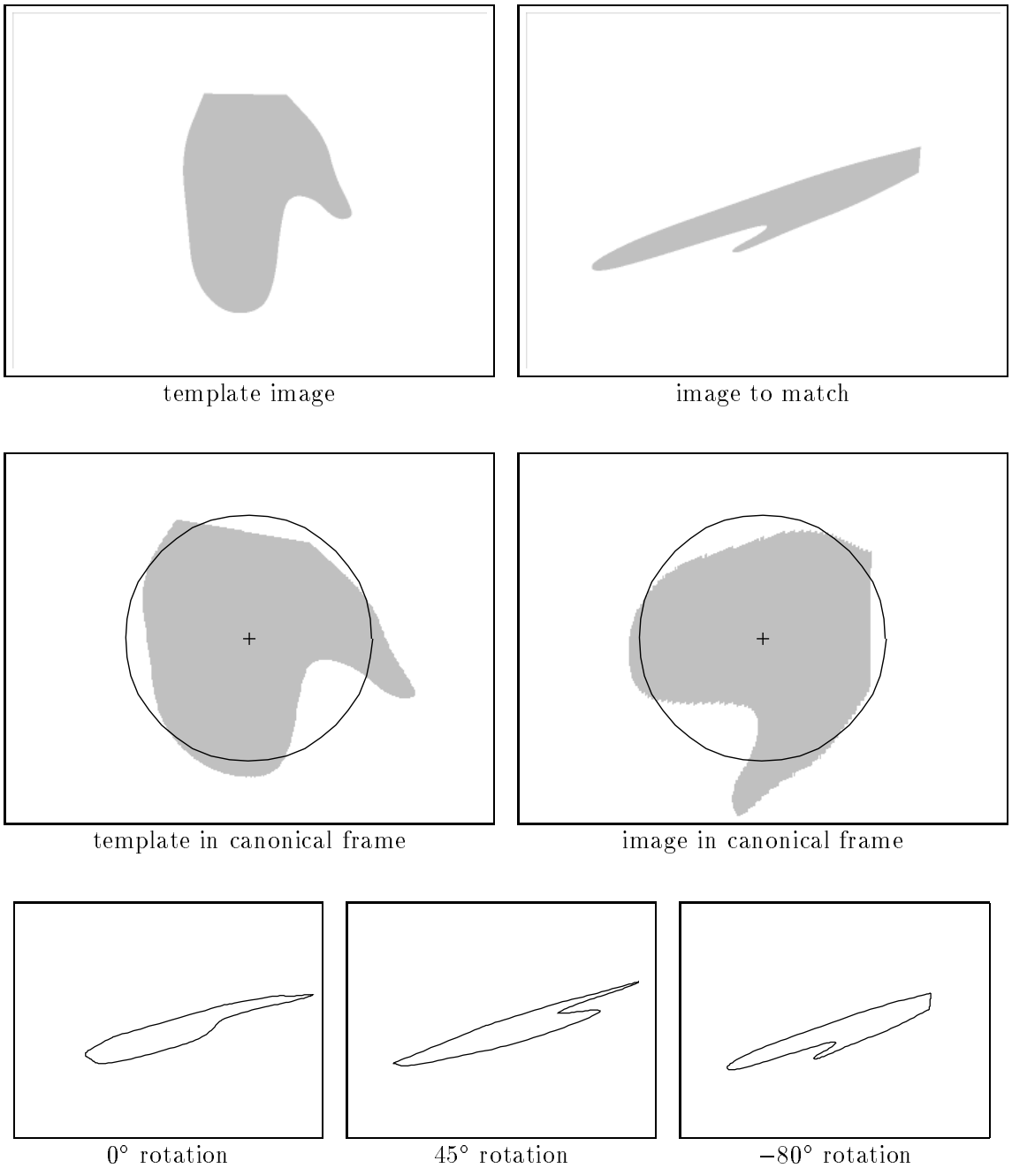


Figure 6.3: **An illustration of the canonical frame** *By computing the first and second moments of the template and the image, it is possible to match five out of the six affine degrees of freedom. The simplest way to see this remaining degree is to show both the template and the image in the canonical frame. This is a frame chosen so that both the template and image have the same first and second moments, and that the second moment matrix of both is the identity. In the figure, the circle drawn on the pictures is the circle with the same first and second moments as the images. In this frame the two are related to each other by an unknown rotation. This rotation is the missing degree of freedom. The bottom row of figures shows the template in the image frame after it has undergone different rotations in the canonical frame. In this case, the missing degree of freedom was* $-80°$.

6.2.6 Statistically approximating the sixth d.o.f. through prior information

In the absence of any additional information that could help resolve the missing degree of freedom, it is reasonable to try and find what the *mean* value of A and its covariance are, and use these to initialise the Kalman filter. Algebraically, this can be written as

$$\mathsf{E}(A) \triangleq \begin{pmatrix} \mathsf{E}(u_x) & \mathsf{E}(u_y) \\ \mathsf{E}(v_x) & \mathsf{E}(v_y) \end{pmatrix} = \int_0^{2\pi} PO(\theta)^T Q^{-1} p(\theta) d\theta, \tag{6.22}$$

$$\mathsf{Var}(A) \triangleq \int_0^{2\pi} \begin{pmatrix} u_x - \mathsf{E}(u_x) \\ u_y - \mathsf{E}(u_y) \\ v_x - \mathsf{E}(v_x) \\ v_y - \mathsf{E}(v_y) \end{pmatrix} \begin{pmatrix} u_x - \mathsf{E}(u_x) \\ u_y - \mathsf{E}(u_y) \\ v_x - \mathsf{E}(v_x) \\ v_y - \mathsf{E}(v_y) \end{pmatrix}^T p(\theta) d\theta. \tag{6.23}$$

This is a slightly non-standard definition of $\mathsf{Var}(A)$, since the variance of A should be a fourth order tensor, but for notational convenience the meaning should be clear.

By making the assumption that all values of θ are equally likely (i.e. $\theta \sim U(0, 2\pi)$), the equations of the expected value of A greatly simplify:

$$\mathsf{E}(A) = \int_0^{2\pi} PO^T Q^{-1} p(\theta) d\theta \tag{6.24}$$

$$= \frac{1}{2\pi} P \left(\int_0^{2\pi} O^T d\theta \right) Q^{-1} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \tag{6.25}$$

$$\mathsf{Var}(A) = \int_0^{2\pi} p(\theta) \begin{pmatrix} u_x - \mathsf{E}(u_x) \\ u_y - \mathsf{E}(u_y) \\ v_x - \mathsf{E}(v_x) \\ v_y - \mathsf{E}(v_y) \end{pmatrix} \begin{pmatrix} u_x - \mathsf{E}(u_x) \\ u_y - \mathsf{E}(u_y) \\ v_x - \mathsf{E}(v_x) \\ v_y - \mathsf{E}(v_y) \end{pmatrix}^T d\theta \tag{6.26}$$

$$= \frac{1}{2\pi} \int_0^{2\pi} \begin{pmatrix} u_x \\ u_y \\ v_x \\ v_y \end{pmatrix} \begin{pmatrix} u_x & u_y & v_x & v_y \end{pmatrix} d\theta. \tag{6.27}$$

So the assumption that $\theta \sim U(0, 2\pi)$ doesn't help recover the sixth degree of freedom (since $\mathsf{E}(A) = \left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right)$ maps the template onto a point), but it does give a covariance matrix for the uncertainty of the parameters which can be used for initialising the state uncertainty of the Kalman filter (chapter 7).

However, in the case that there is a prior for A (potentially obtained from any of the methods described in chapter 7), it is possible to get an estimate of the sixth degree of

freedom either by computing the expected value of A or by sampling through all the values of θ, computing A for each value and choosing the *maximum a priori* value of A. Figure 6.4 shows the result of such a method (which is discussed in more detail in section 6.2.11). Training data for lip tracking revealed that v_x is always very small. By setting $v_x = 0$, it is possible to get good initialisation without having to measure the sixth degree of freedom in the image.

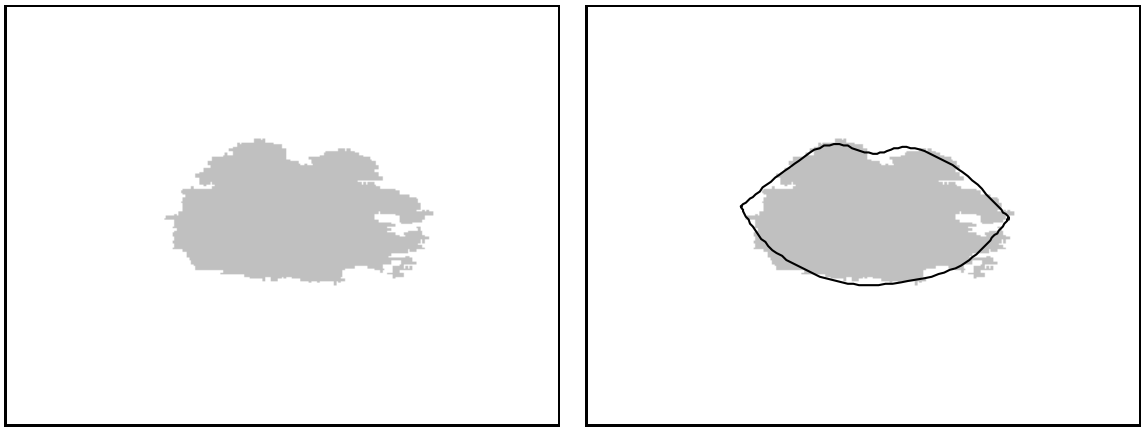


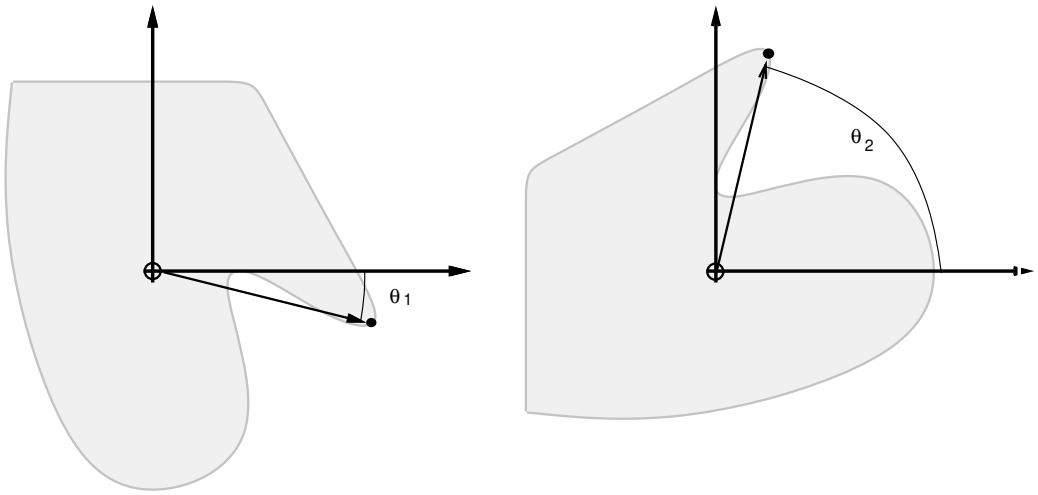
image to match 'best' five d.o.f. solution

Figure 6.4: **Recovering the sixth affine degree of freedom from prior knowledge** *It is possible to recover the sixth affine degree of freedom by using prior knowledge about A. A PCA on all of the A matrices obtained from a training sequence of lip tracking showed that v_x (the lower left corner of A) varied little from zero. So, the best five degree approximation to the true affine deformation will have $v_x = 0$. To find the affine deformation that satisfies this constraint, set $l = 0$ and L to have a 1 in the lower left corner and zero in all others. Then substitute into equation 6.45 and solve for θ and subsequently A. The solution is shown above and fits well because A has a strong prior. Other systems may not have such strong priors. For more detail, see section 6.2.11.*

6.2.7 Second order moments and a distinguished point

In the case when a distinguished point is known in both the template and the image being used for initialisation, it is trivial to recover the sixth degree of freedom. Following the example of Koenderink in his image flow paper (Koenderink, 1986), the easiest way is to transform both the image and the template into the canonical frame, as in figure 6.5. Next, the distinguished point is found in both. By converting the location of the distinguished

point into polar coordinates (r, θ_1) and (r, θ_2), it is possible to find $\theta = \theta_2 - \theta_1$, the missing degree of freedom. Note that since in the canonical frame the image and template differ only by a rotation, r is the same in both the image and template.



template image image to match

Figure 6.5: **An illustration of the distinguished point method** *It is possible to recover the sixth affine degree of freedom by locating a distinguished point on both the template and the image (in this case, the tip of the thumb). Once the image and template are transformed into the canonical frame, the angle can be found between the x-axis and the distinguished point in each. The difference of the two angles is the missing rotation.*

6.2.8 Backprojection of fit for sixth d.o.f.

In section 6.2.6, information from a prior was used to provide statistical information that was used to recover the sixth degree of freedom. A second general source of information is from the initialisation image itself. Each of the deformations in the family of solutions can be checked on the image for the amount of "support" that the image gives it. Then, the deformation with the most support is chosen as the correct value for initialisation. In general there are two kinds of support that can be used, support based on area in common

and support based on edge information.

Computing the amount of support based on area in common is basically counting the number of pixels interior of the template and on the image. Normally it is difficult to define a good metric based on this kind of method, because the metric would have to handle the case when one object was of a much larger area than the other. In such a case the larger object could completely obscure the smaller, even though the fit would be poor.

Fortunately, the moment-based initialiser helps solve this problem. Because the two objects being compared are guaranteed to have the same area, a much simpler metric can be used. In this case, it is possible to use $\|I_1 \cap I_2\|$, i.e. count the number of pixels that are on the interior of both the template and the image. On both synthetic and real images this method is quite successful, but it requires comparing a large number ($\mathcal{O}(c_0)$) of pixels which makes it quite costly.

A second metric of "goodness of fit" is to search normally from the template to find its intersection with the edge of object in the image. Since the initialiser is for use with a contour-based tracker, this would seem to be the most logical choice of metric. It also has the added benefit of being much less computationally expensive than the area method. Tables 6.1 and 6.2 show that in general both of these methods are successful at recovering the sixth degree of freedom in synthetic and real images. Figures 6.6 through 6.11 show pictures of the two methods successfully initialising on synthetic images of various kinds with and without noise, as well as on real images.

6.2.9 Using higher order moments

It is also possible to resolve this last degree of freedom by comparing higher-order moments of the two systems. This can be hard to solve analytically because it is a non-linear system. However, the closed-form solution in equation 6.17 makes things slightly simpler.

Initialisation using the second-order moments as described in section 6.2.4 gives 5 d.o.f. where the remaining degree is a rotation in a special basis (the canonical frame). The

synthetic images, no noise added

	Estimated angle (in radians)								
Image	$\cos^{-1}\sqrt[3]{a_1}$	$\sin^{-1}\sqrt[3]{a_4}$	$\tan^{-1}\frac{a_2}{a_1}$	$\tan^{-1}\frac{a_3}{a_2}$	$\tan^{-1}\frac{a_4}{a_3}$	$\tan^{-1}\frac{a_2+a_4}{a_1+a_3}$	normal dist.	max. area	ground truth
hand1	1.70	-1.35	-1.66	1.71	-1.40	-1.41	-1.37	-1.37	-1.41
lips1	1.35	undef	1.23	-0.76	-1.60	-1.59	-1.57	-1.57	-1.59
plant1	1.32	-1.43	0.198	-1.46	-1.60	-1.58	-1.60	-1.60	-1.58

synthetic images, noise added

	Estimated angle (in radians)								
Image	$\cos^{-1}\sqrt[3]{a_1}$	$\sin^{-1}\sqrt[3]{a_4}$	$\tan^{-1}\frac{a_2}{a_1}$	$\tan^{-1}\frac{a_3}{a_2}$	$\tan^{-1}\frac{a_4}{a_3}$	$\tan^{-1}\frac{a_2+a_4}{a_1+a_3}$	normal dist.	max. area	ground truth
hand2	1.11	undef	-0.769	2.21	-1.49	-1.43	-1.40	-1.44	-1.41
lips2	undef	undef	2.74	-0.48	-1.90	-2.96	-1.52	-1.55	-1.59
plant2	0.86	-0.94	0.24	-0.075	-1.58	-1.04	-1.60	-1.60	-1.58

Table 6.1: **A numerical comparison using synthetic images of different methods of estimating the sixth affine degree of freedom** *By comparing the first and second moments of a template and an affine deformation of that image, it is possible to recover five out of the six affine degrees of freedom. There are several ways to recover the sixth degree of freedom. Equation 6.33 specifies several different estimates of θ. On noiseless images,* $\tan^{-1}\frac{a_2+a_4}{a_1+a_3}$ *turns out to be a good estimator of θ. The other estimates are less reliable. However, all estimators based on the third moments are sensitive to noise. 'undef' signifies that θ was complex by a given estimate. It is also possible to recover the sixth degree of freedom by backprojecting a large sample of the family of solutions onto the original image and choosing the sample that either minimises normal distance ('normal dist.') or the common area ('max. area') between the sample and the image. On noisy synthetic images, both of these two methods performed more reliably than any of the moment-based methods. For pictures of the solutions, see figures 6.6 and 6.7*

real images

	Estimated angle (in radians)							
Image	$\cos^{-1}\sqrt[3]{a_1}$	$\sin^{-1}\sqrt[3]{a_4}$	$\tan^{-1}\frac{a_2}{a_1}$	$\tan^{-1}\frac{a_3}{a_2}$	$\tan^{-1}\frac{a_4}{a_3}$	$\tan^{-1}\frac{a_2+a_4}{a_1+a_3}$	normal dist.	max. area
real hand 1	undef	0.67	0.04	0.79	1.33	0.22	0.25	0.22
real hand 2	undef	-0.73	-0.39	2.31	-0.48	-0.42	-0.39	-0.44
real hand 3	1.11	-0.30	1.18	0.46	-0.23	0.77	0.59	0.44
real plant 1	undef	-0.89	0.04	1.36	-1.01	-0.23	-1.64	-1.69
real plant 2	undef	0.62	2.90	-1.16	2.82	2.86	-1.67	-1.62
real plant 3	undef	-0.83	3.05	0.06	-1.56	-2.86	-0.98	-1.05
real plant 4	undef	0.94	3.00	-1.19	2.34	2.77	-1.54	-1.60
real plant 5	2.49	-1.34	-2.64	-2.33	-1.87	-2.15	2.18	-0.12
real plant 6	undef	undef	0.22	1.07	0.95	0.51	0.71	0.71

Table 6.2: **A numerical comparison using real images of different methods of estimating the sixth affine degree of freedom.** *When using noisy synthetic images (table 6.1), it was found that backprojecting and using the 'minimum normal distance' or 'maximum common area' methods gave reliable estimates of the sixth affine degree of freedom, whereas methods based on the third moments were much worse. When real images are used, the same result holds. The backprojection methods give estimates of θ that subjectively look to be much better fits than those of given by the third moments. Of the third moments methods,* $\tan^{-1}\frac{a_2+a_4}{a_1+a_3}$ *is the most reliable estimator. For pictures of the solutions, see figures 6.8, 6.9, 6.10 and 6.11.*

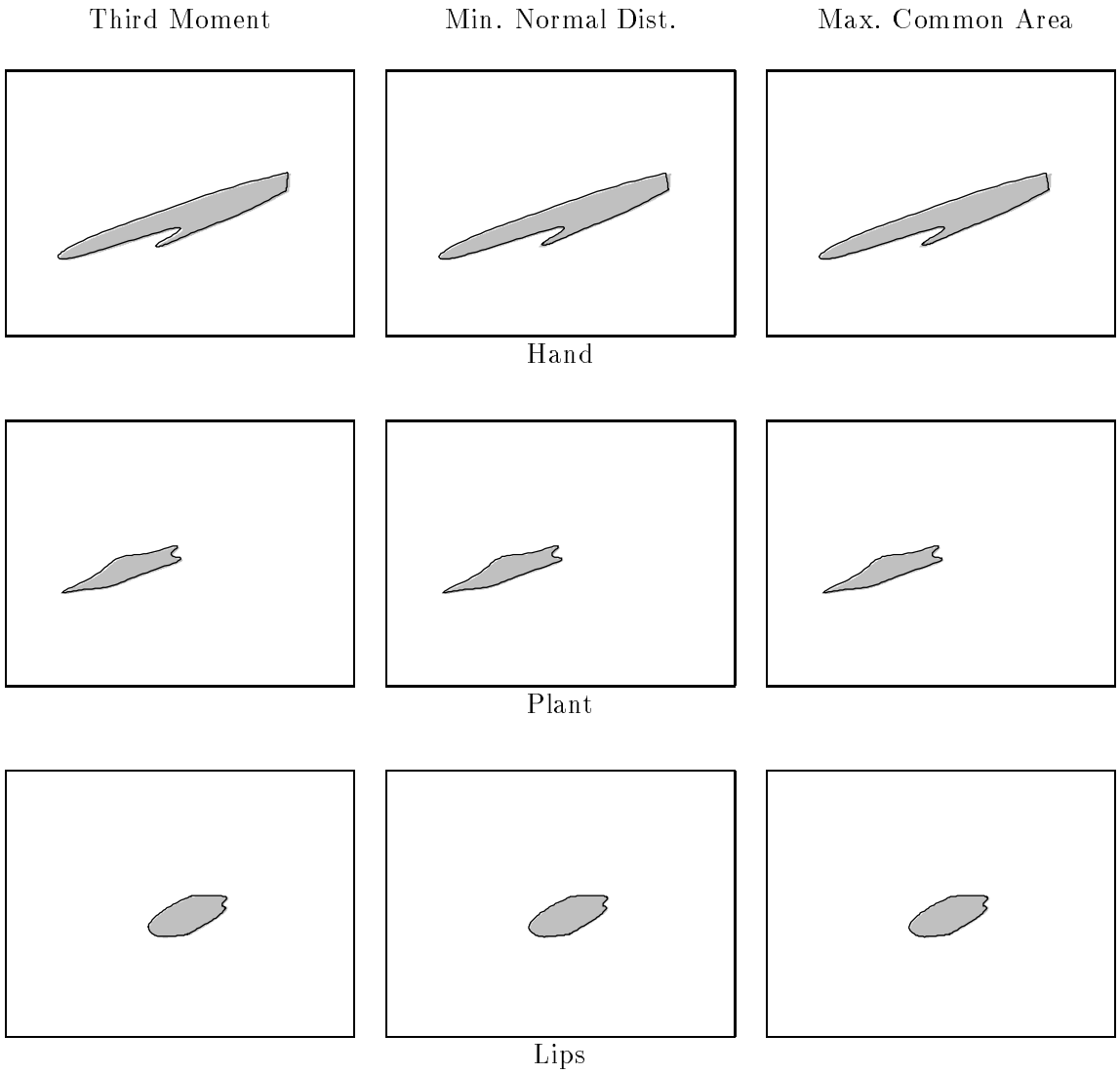


Figure 6.6: **A visual comparison using noiseless synthetic images of different methods of estimating the sixth affine degree of freedom.** *When presented with images that are exact affine deformation of the template, it is possible to recover five of the six affine degrees of freedom by comparing the first and second moments of the template and the image (section 6.2.4). There are three different methods to recover the sixth degree of freedom. In each image shown, the grey area is the image and the black outline is the template. All three methods perform well on synthetic images, though for the hand image, the third moment method has a slightly better fit on the upper right corner of the hand.*

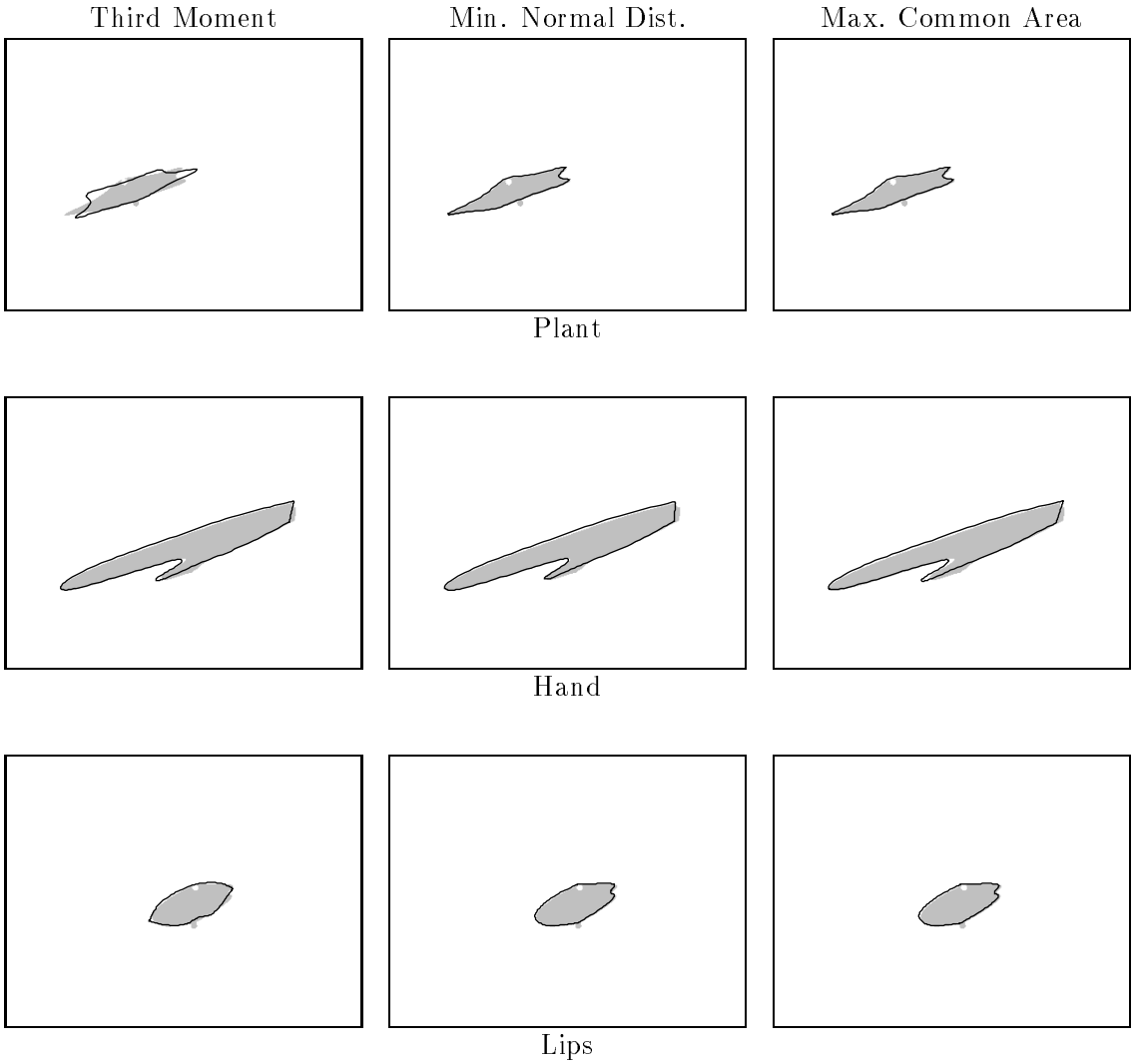


Figure 6.7: **The recovery of θ from third moments is sensitive to noise in synthetic images.** *Three methods of initialising from moments are compared on synthetic images that have had noise added to them. The images (grey area) are all affine deformations of the template image (black line), and five out of the six degrees of freedom have been recovered using the canonical frame method described in section 6.2.4 and illustrated in figure 6.3. The sixth degree of freedom is then recovered using three different methods. The 'third moment' method uses equation 6.34 and the* $\theta = \tan^{-1}\left(\frac{a_2+a_4}{a_1+a_3}\right)$ *(see table 6.1). On both the images of plants and lips, the rotation is computed incorrectly. The 'minimal normal displacement' and 'maximum area' methods (described in section 6.2.8) both perform better and are not distracted by the added noise.*

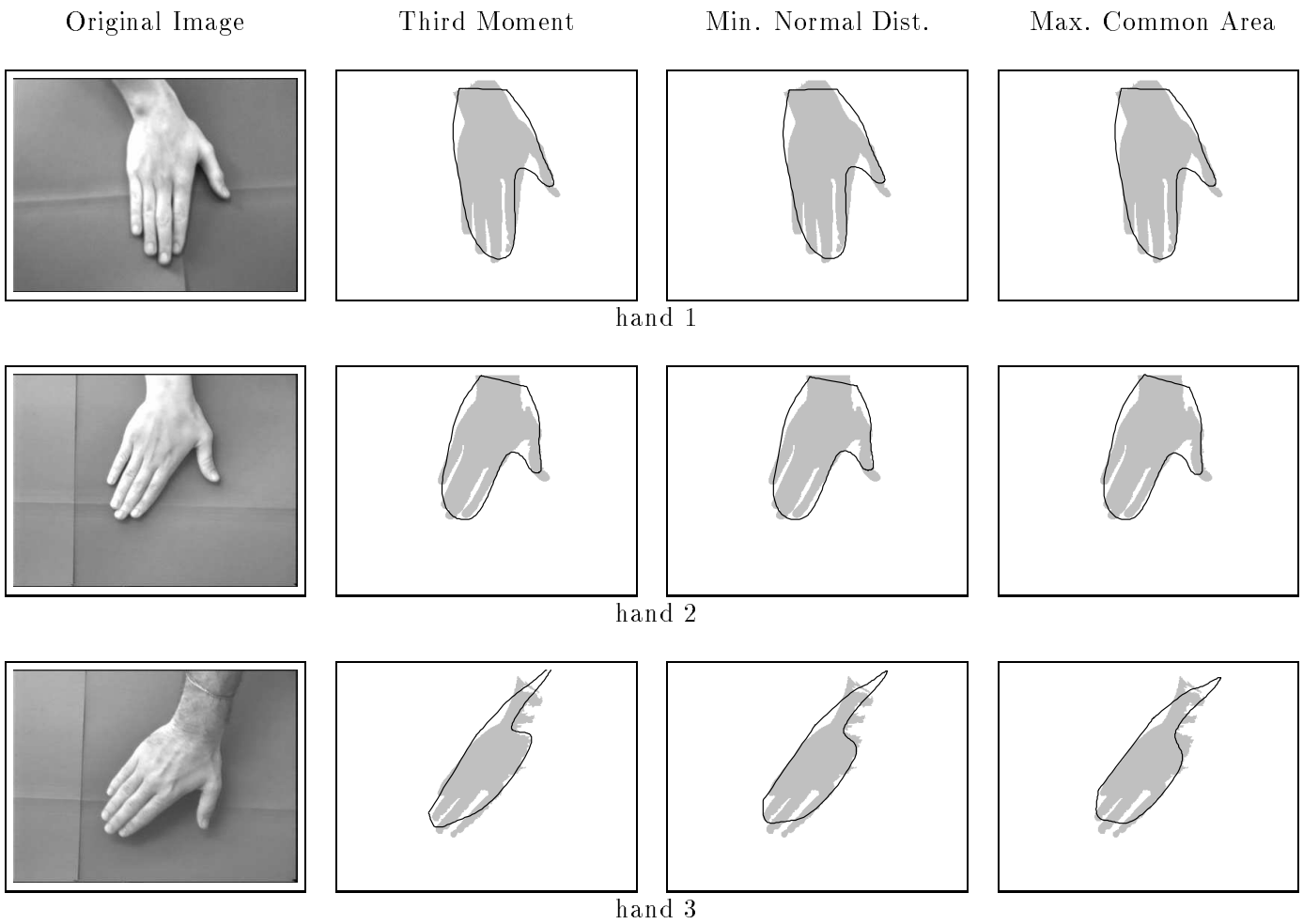


Figure 6.8: **A visual comparison using hand images of different methods of estimating the sixth affine degree of freedom.** *Three images of hands are classified using intensity thresholding with the threshold being chosen by the Kapur method (Kapur et al., 1985). Initialisation is done as in figures 6.6 and 6.7. Segmentation is good on the top two images, and all three methods perform well. On the bottom image, the thumb has been misclassified as background. The centroid of the hand is still fairly well localised but the rotation is wrong and tracking will likely fail.*

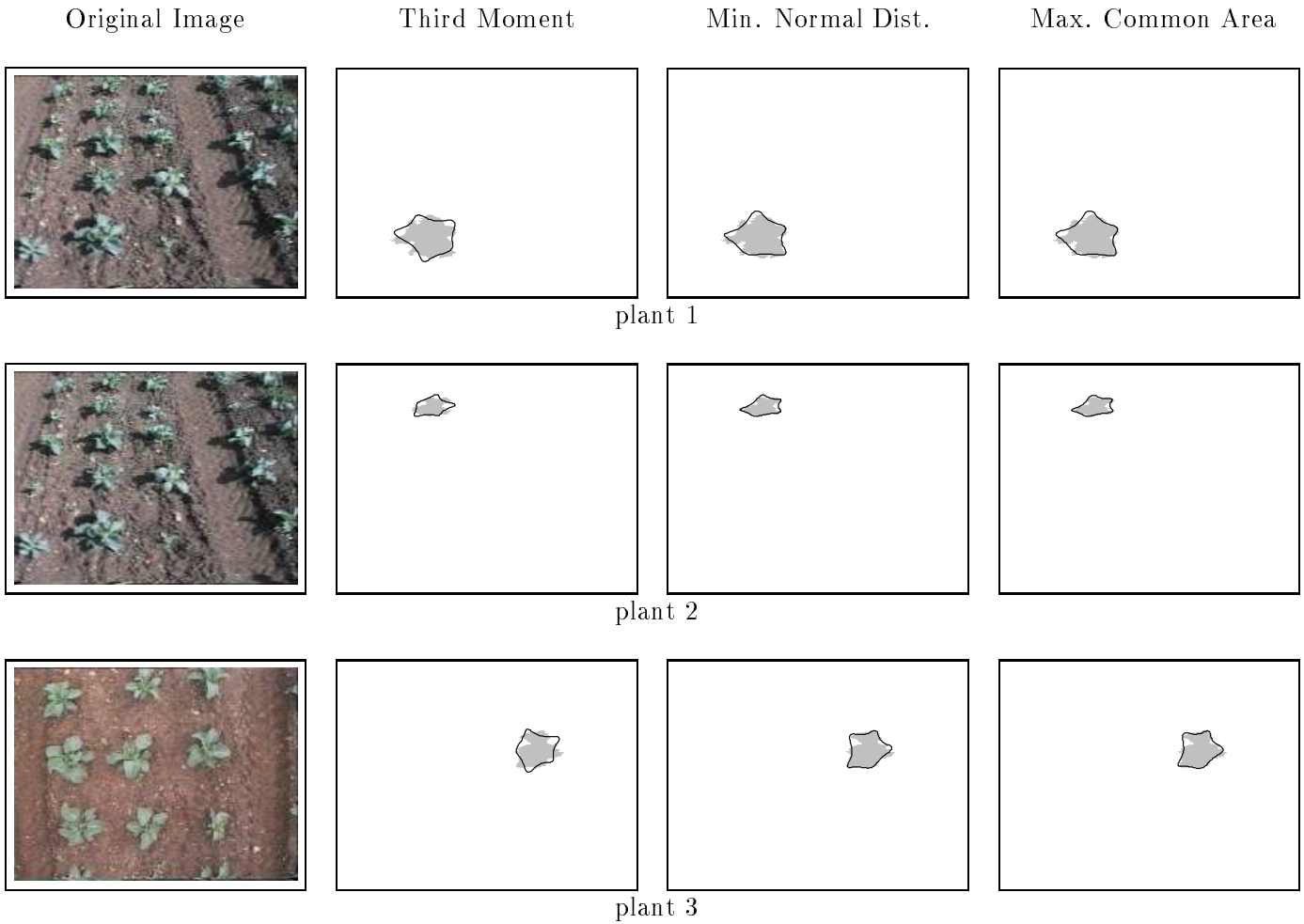


Figure 6.9: **A visual comparison using some plant images of different methods of estimating the sixth affine degree of freedom.** *These images of plants were thresholded based on their hue by a Kapur thresholder, then automatically initialised as in section 6.2.4. All three methods accurately localised the plants but the third moment method gives a worse fit than either of the other two.*

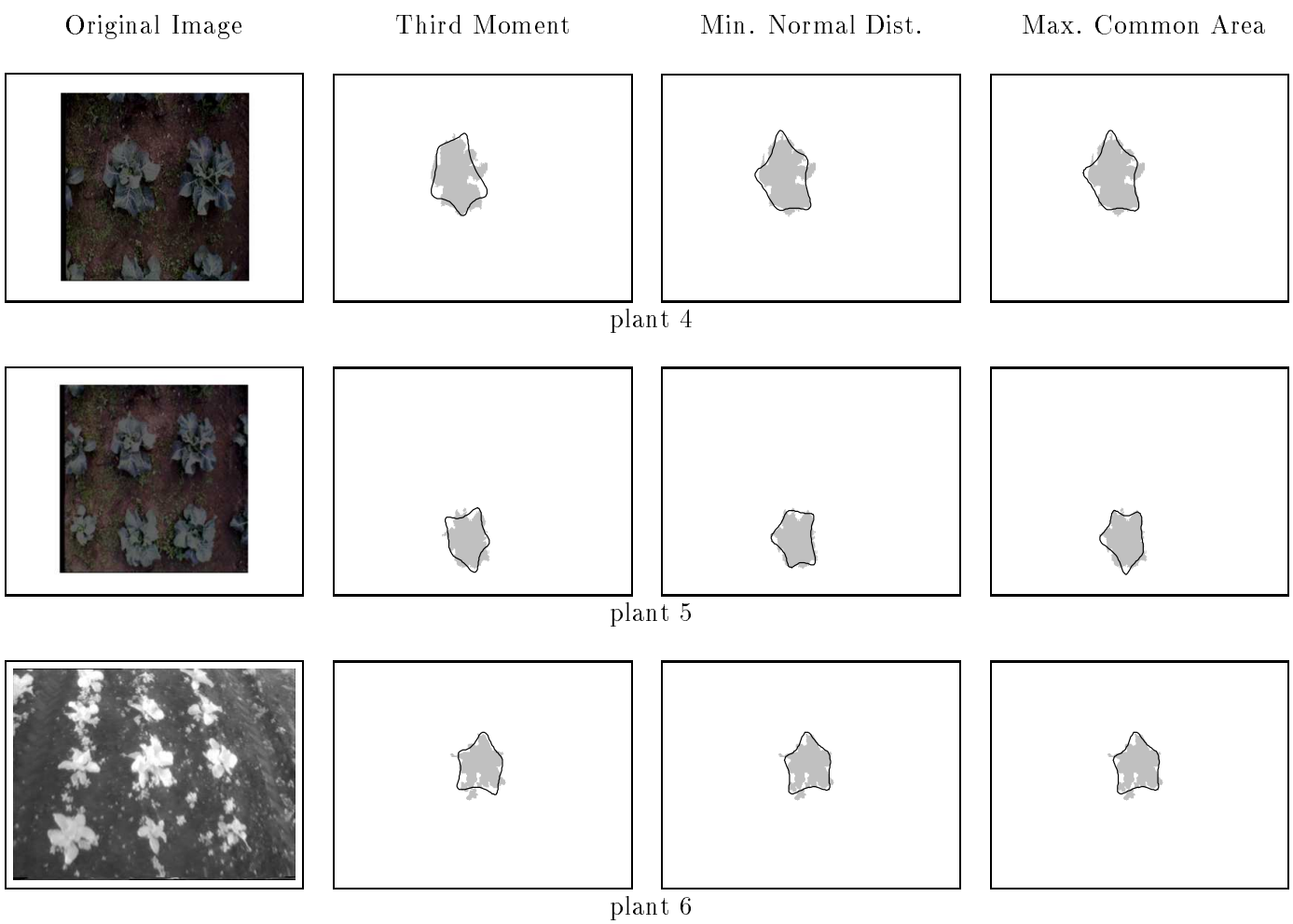


Figure 6.10: **Another visual comparison using plant images of different methods of estimating the sixth affine degree of freedom.** *All three images were Kapur thresholded (the top two based on their hue, the bottom by intensity) and initialised as in figure 6.6. As before, the minimum normal distance and maximum common area methods subjectively give equally good fits, but the fit from the third moment method is inferior.*

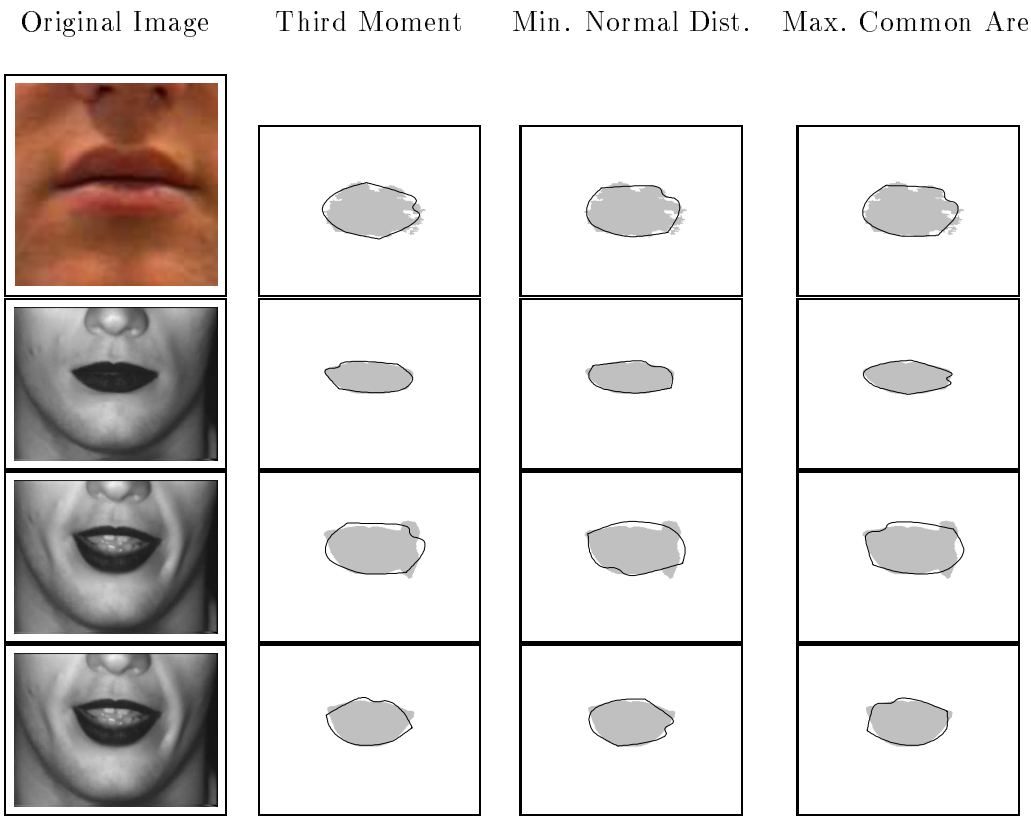


Figure 6.11: **A visual comparison using lip images of different methods of estimating the sixth affine degree of freedom.** *The top image is thresholded by its hue and the middle two by their intensity. Thresholds for all three are determined by the Kapur method. The bottom image has been segmented by manually choosing a threshold. All four are initialised as in figure 6.6. While the initialisation method can accurately fit five of the six affine degrees of freedom, none of the methods can reliably recover the remaining rotation.*

transformations required to put the systems in this canonical frame were specified by the matrices P and Q. Once the shape on the screen and the template have been transformed into the canonical frame, the shape from the screen becomes a rotation of the template shape. To recover the sixth degree of freedom we only need to find out which one. By comparing the third order moments for both systems in this canonical frame, we can obtain a simple relationship that will enable us to recover the remaining rotation.

The third order moments are defined like this:

$$c_{xxx} \triangleq \frac{1}{c_0}\iint (x - c_x)^3\, dx\, dy \tag{6.28}$$

$$c_{xxy} \triangleq \frac{1}{c_0}\iint (x - c_x)^2 (y - c_y)\, dx\, dy \tag{6.29}$$

$$c_{xyy} \triangleq \frac{1}{c_0}\iint (x - c_x)(y - c_y)^2\, dx\, dy \tag{6.30}$$

$$c_{yyy} \triangleq \frac{1}{c_0}\iint (y - c_y)^3\, dx\, dy. \tag{6.31}$$

In the canonical frame, the two systems are related by a simple rotation, so

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{6.32}$$

By substituting equation 6.32 into equations 6.28 through 6.31 (remembering that the Jacobian of a rotation matrix is 1, and so it doesn't enter into the equation) the following relation is obtained:

$$\begin{pmatrix} c'_{xxx} \\ c'_{xxy} \\ c'_{xyy} \\ c'_{yyy} \end{pmatrix} = \begin{pmatrix} c_{xxx} & 3c_{xxy} & 3c_{xyy} & c_{yyy} \\ c_{xxy} & -c_{xxx} + 2c_{xyy} & -2c_{xxy} + c_{yyy} & -c_{xyy} \\ c_{xyy} & -2c_{xxy} + c_{yyy} & c_{xxx} - 2c_{xyy} & c_{xxy} \\ c_{yyy} & -3c_{xyy} & 3c_{xxy} & -c_{xxx} \end{pmatrix} \begin{pmatrix} \cos^3\theta \\ \cos^2\theta \sin\theta \\ \cos\theta \sin^2\theta \\ \sin^3\theta \end{pmatrix}. \tag{6.33}$$

Once this is obtained, the matrix can be inverted giving us the different combinations of $\sin\theta$ and $\cos\theta$. By taking almost any two of these it is possible to *uniquely* recover θ. One important point is that the matrix to be inverted consists *only* of the moments of the template, thus all its values and the inverse itself can be computed offline, limiting the online computations to two dot products.

Unfortunately, the estimates of the third order moments are quite sensitive to noise in the image and the estimate of the lower order moments. Table 6.1 shows that in synthetic

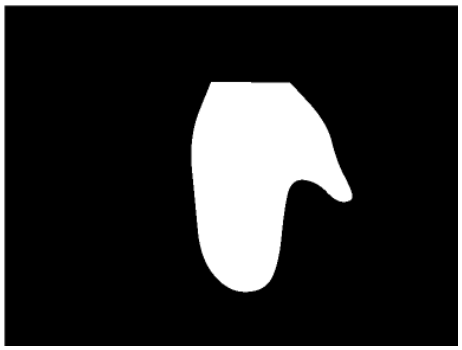
images without noise, third order moments are capable of recovering the sixth degree of freedom. However figure 6.12 shows that in the presence of noise, the estimates of the third moments are very poor, and table 6.2 shows that the estimate of θ is poor for most combinations of estimates. One particular combination of the estimates proves to be much more reliable than the rest. For simplicity, we assign variable names to the various estimates of the functions of θ.

$$\begin{pmatrix} c_{xxx} & 3c_{xxy} & 3c_{xyy} & c_{yyy} \\ c_{xxy} & -c_{xxx}+2c_{xyy} & -2c_{xxy}+c_{yyy} & -c_{xyy} \\ c_{xyy} & -2c_{xxy}+c_{yyy} & c_{xxx}-2c_{xyy} & c_{xxy} \\ c_{yyy} & -3c_{xyy} & 3c_{xxy} & -c_{xxx} \end{pmatrix}^{-1} \begin{pmatrix} c'_{xxx} \\ c'_{xxy} \\ c'_{xyy} \\ c'_{yyy} \end{pmatrix} \triangleq \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} \tag{6.34}$$

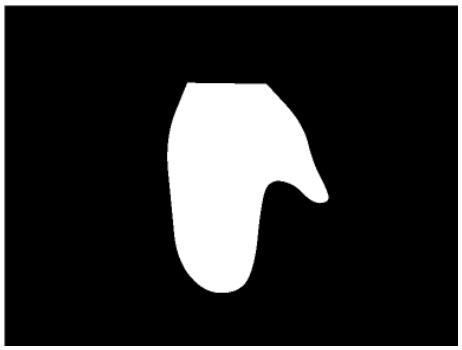
Experimentally, it turns out that $\theta = \tan^{-1}\left(\frac{a_2+a_4}{a_1+a_3}\right)$ is a fairly reliable estimator of θ. Tables 6.1 and 6.2 compare the different estimates of θ in several test cases. But even when $\theta = \tan^{-1}\left(\frac{a_2+a_4}{a_1+a_3}\right)$ is used, the third moments consistently give a worse estimate than the normal distance and maximum area methods (see figures 6.6 through 6.11 for a visual comparison of the three methods).

Of course it would be preferable to use information from $a_1, \ldots a_4$ to come up with an improved solution. All four are functions of θ, so the system is overconstrained. A standard method for solving such overconstrained systems is Least Squares Estimation (Press et al., 1988). In the case of a linear system, it is easy to minimise the squared error using either the Moore-Penrose (pseudo-inverse) (Barnett, 1990) or singular value decomposition (Press et al., 1988). However since $a_1, \ldots a_4$ are non-linear functions of θ, such methods cannot be directly applied.

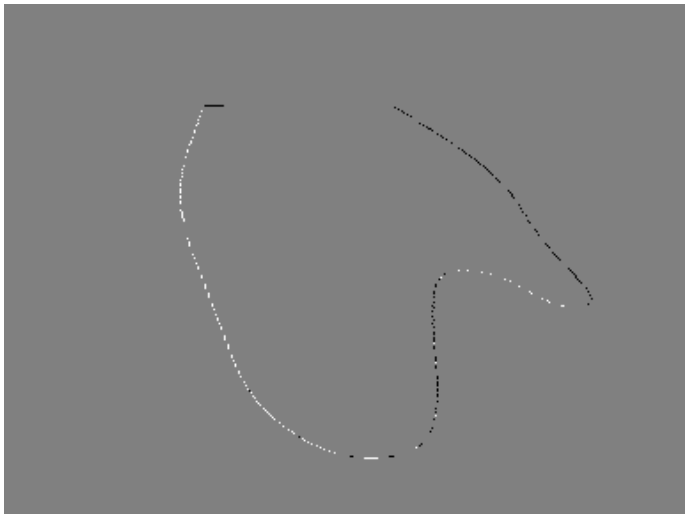
The Levenberg-Marquardt method is the standard method for solving overconstrained nonlinear systems using least-squares (Press et al., 1988). Levenberg-Marquardt models the error function as a second-order Taylor series (i.e. a quadratic function), then minimises this function through gradient descent. By using certain properties of the error function, the minimum error can be found in relatively few iterations.



(a)



(b)



(c)

Moment	Error (percent)
c_{xxx}	-7.5
c_{xxy}	-0.511
c_{xyy}	9.27
c_{yyy}	2.18

(d)

Method of estimating θ	Error (radians)
$\cos^{-1}\sqrt[3]{a_1}$	undef
$\tan^{-1} a_2/a_1$	0.018
$\tan^{-1} a_3/a_2$	0.853
$\tan^{-1} a_4/a_3$	1.224
$\tan^{-1}(a_2+a_4)/(a_1+a_3)$	0.075
$\sin^{-1}\sqrt[3]{a_4}$	0.406

(e)

Figure 6.12: **Third-order moments are very sensitive to noise** *Figures 6.12(a) and 6.12(b) show two images of a hand translated relative to each other. Figure 6.12(c) shows the difference of the images after they've been transformed into the canonical frame. Grey pixels indicate agreement between the two images, and white pixels indicate pixels on in the left image and not on the right. Black pixels are the opposite. Even though the correspondence is very good (less than a pixel error), the errors in the third moments are substantial (figure 6.12(d)) and the errors in the estimate of θ are worse (figure 6.12(e)). In the case of $\cos^{-1}\sqrt[3]{a_1}$, it was not possible to get an estimate of θ because $a_1 > 1$. For the definition of $a_1, \ldots a_4$ see equation 6.34.*

Maximum likelihood estimation (MLE) can also be used to recover a more accurate value of θ. By viewing $a_1, \ldots a_4$ as functions of samples of a random variable parameterised by θ, it is possible to compute the likelihood that $a_1, \ldots a_4$ were generated by a given θ. The θ that maximises this likelihood can then be chosen as the best estimate.

However, both Levenberg-Marquardt and MLE require estimates of the error of $a_1, \ldots a_4$. For linear least-squares, this estimate of error is usually overlooked – it is sufficient to say that all the samples are independent and have the same Gaussian error distribution. It is highly unlikely either is true in equation 6.34.

The error estimates can be obtained in a couple different ways. First, a perturbation analysis can be applied to equation 6.34: by observing how noise added to the estimates of the third-order moments affects the solution, it may be possible to obtain error estimates on $a_1, \ldots a_4$. Secondly, the variance of these terms can be learned through sample images in a similar manner to chapter 5. By computing the third moments on a large number of images and comparing the estimates of $a_1, \ldots a_4$ to the real θ (estimated by hand or through some other method) error estimates can be obtained. Future research will investigate which of these two methods are the most accurate and appropriate.

6.2.10 Reduced dimensionality solutions

Instead of trying to recover all six affine degrees of freedom, it is sometimes possible to restrict the general affine deformation to lie in a lower dimensional subspace of the affine space. One such subspace that is simple to represent is one that allows translation and independent scaling in the x and y directions. This is the subspace of the affine space that restricts the A matrix to be of the form $A = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$ (independent x and y scaling ignoring all rotations and shears). In this case it is particularly easy to recover a and b from the first and second-order moments:

$$\begin{pmatrix} c'_{xx} \\ c'_{yy} \end{pmatrix} = \begin{pmatrix} a^2 c_{xx} \\ b^2 c_{yy} \end{pmatrix} \tag{6.35}$$

This subspace has been used successfully for the initialisation of the plant tracker and is well-suited to the general problem of initialising systems known to have very little rotation.

A more common subspace to use is the family of similarity transformations (translation plus rotation and zoom). Similarities are a subspace of affine transformations that correspond to restricting the A matrix to be of the form $A = \begin{pmatrix} r\cos\theta & r\sin\theta \\ -r\sin\theta & r\cos\theta \end{pmatrix}$. Once again, it is easy to recover r and θ from the first and second moments:

$$C' = \begin{pmatrix} r\cos\theta & r\sin\theta \\ -r\sin\theta & r\cos\theta \end{pmatrix} C \begin{pmatrix} r\cos\theta & -r\sin\theta \\ r\sin\theta & r\cos\theta \end{pmatrix} \tag{6.36}$$

$$\det C' = r^4 \det \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \det C \det \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

$$\det C' = r^4 \det C \tag{6.37}$$

It is also possible to recover r from equation 6.14 which should be less sensitive to noise in the image. To solve for θ, expand out equation 6.36 to obtain

$$c'_{xx} - c'_{yy} = r^2 \left((c_{xx} - c_{yy}) \cos 2\theta + 2c_{xy} \sin 2\theta \right) \tag{6.38}$$

By substituting $u = \tan\theta$, you get

$$\begin{aligned} c'_{xx} - c'_{yy} &= r^2 \left((c_{xx} - c_{yy}) \frac{1-u^2}{1+u^2} + 2c_{xy} \frac{2u}{1+u^2} \right) \\ (1+u^2)(c'_{xx} - c'_{yy}) &= r^2 \left((c_{xx} - c_{yy})(1-u^2) + 4c_{xy}u \right) \\ \tan\theta &= \frac{2r^2 c_{xy} \pm \sqrt{-(c'_{xx} - c'_{yy})^2 + r^4 (c_{xx} - c_{yy})^2 + 4c_{xy}^2 r^4}}{c'_{xx} - c'_{yy} + r^2 c_{xx} - r^2 c_{yy}}. \end{aligned} \tag{6.39}$$

6.2.11 Optimal subspaces within affine spaces

Sections 6.2.7 through 6.2.10 discussed various methods of recovering the sixth degree of freedom through additional measurements on the image (distinguished points, additional moments, etc.). An alternative is to use prior information from training sequences to estimate A without additional measurements. In section 6.2.6 it was postulated that a prior on A could help recover the sixth degree of freedom. Of course the prior can't measure the sixth degree of freedom, but it can help choose the A matrix that statistically will have the least error.

One of the basic assumptions of the Kalman filter is that the state at any time is a Gaussian random variable, parametrised by a mean and variance. The prior of A is the same as the state estimate and uncertainty of the state of the Kalman filter at time zero. There are several different ways that this can be estimated, and any of the methods in chapter 7 can be used. The variance of the prior can be decomposed using an eigenvector/eigenvalue decomposition to obtain an orthogonal basis $L_{max}, \ldots, L_{min}$ where L_{max} is the eigenvector with the largest eigenvalue. With this basis, any matrix A can be rewritten as $l_{max}L_{max} + \ldots + l_{min}L_{min}$ where L_{max} is the component of A shown to have the most variation, L_{min} has the least and the mean values $\hat{l}_{max} = \mathsf{E}(l_{max}), \ldots, \hat{l}_{min} = \mathsf{E}(l_{min})$ are known.

This decomposition of the prior of A is equivalent to a principal components analysis (PCA) on the non-translational components of the affine space. Motion in direction L_{max} will typically be larger than motion in all other directions. Similarly, motion in direction L_{min} will typically be smaller than motion in all other directions. In other words, in the absence of all other information, guessing that $\hat{l}_{min} = l_{min}$ will be a better estimate than any other.

This observation can be used to initialise the sixth degree of freedom from prior information. From the first and second-order moments, five out of the six affine degrees of freedom are known. The remaining degree of freedom is parametrised by the matrix $O^T(\theta)$. The problem of choosing the best A given its prior is reduced to choosing a θ such $A \cdot L_{min} = \hat{l}_{min}$ where

$$A \cdot L \triangleq \sum_{i,j=1}^{2} A_{ij} L_{ij}, \tag{6.40}$$

which is a 'dot-product' operation for matrices.

Given a P and Q^{-1} such that $A = PO^TQ^{-1}$, the effects of $\sin\theta$ and $\cos\theta$ can be

recovered as follows:

$$
\begin{aligned}
A &= PO^TQ^{-1} = P\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} Q^{-1} && (6.41)\\
&= P\left(\begin{pmatrix} \cos\theta & 0 \\ 0 & \cos\theta \end{pmatrix} + \begin{pmatrix} 0 & -\sin\theta \\ \sin\theta & 0 \end{pmatrix}\right) Q^{-1} && (6.42)\\
&= P\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} Q^{-1}\cos\theta + P\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} Q^{-1}\sin\theta && (6.43)\\
&= PQ^{-1}\cos\theta + P\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} Q^{-1}\sin\theta && (6.44)
\end{aligned}
$$

Setting $A \cdot L_{min} = \hat{l}_{min}$, θ can be recovered:

$$
\begin{aligned}
\hat{l}_{min} = A \cdot L_{min} &= \left(PQ^{-1}\cos\theta + P\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} Q^{-1}\sin\theta\right) \cdot L_{min} && (6.45)\\
&= \left(PQ^{-1}\cos\theta\right) \cdot L_{min} + \left(P\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} Q^{-1}\sin\theta\right) \cdot L_{min} && (6.46)\\
&= \cos\theta(PQ^{-1} \cdot L_{min}) + \sin\theta(P\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} Q^{-1} \cdot L_{min}). && (6.47)
\end{aligned}
$$

By defining $\Upsilon \triangleq PQ^{-1} \cdot L_{min}$ and $\Phi \triangleq P\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} Q^{-1} \cdot L_{min}$ and $u = \tan\frac{\theta}{2}$,

$$
\begin{aligned}
\hat{l}_{min} &= \Upsilon\cos\theta + \Phi\sin\theta \\
&= \frac{1-u^2}{1+u^2}\Upsilon + \frac{2u}{1+u^2}\Phi \\
(1+u^2)\hat{l}_{min} &= (1-u^2)\Upsilon + 2u\Phi \\
\tan\frac{\theta}{2} &= \frac{\Phi \pm \sqrt{-\hat{l}_{min}^2 + \Phi^2 + \Upsilon^2}}{\hat{l}_{min} + \Upsilon} && (6.48)
\end{aligned}
$$

which has been solved using the closed-form solution of the quadratic equation.

Table 6.3 summarises the results from the various reduced dimensionality solutions.

6.2.12 Invariant initialisation

In recent years, a large amount of literature has developed around the use of 'invariants' for pose recovery and model-based object recognition. For certain classes of deformations (namely projective and affine deformations) these methods have been shown to be robust and efficient. (Rothwell et al., 1992; Zisserman A. et al., 1995)

Name	W space	equations
Independent scaling	$\left(a_1 \begin{pmatrix} X \\ 0 \end{pmatrix} + a_2 \begin{pmatrix} 0 \\ Y \end{pmatrix} \right)$	$\begin{pmatrix} c'_{xx} \\ c'_{yy} \end{pmatrix} = \begin{pmatrix} a_1^2 c_{xx} \\ a_2^2 c_{yy} \end{pmatrix}$
Scaling and rotation	$\left(r\cos\theta \begin{pmatrix} X \\ Y \end{pmatrix} + r\sin\theta \begin{pmatrix} -Y \\ X \end{pmatrix} \right)$	$\begin{aligned} rc_0 &= c'_0 \\ c_{xy}\sin 2\theta &= c'_{xx} + c'_{yy} - c_{xx} - c_{yy} \end{aligned}$
Arbitrary 5 d.o.f.	Affine with $A \cdot L_{min} = \hat{l}_{min}$	$\begin{aligned} \Upsilon &\triangleq PQ^{-1} \cdot L_{min} \\ \Phi &\triangleq P \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} Q^{-1} \cdot L_{min} \\ \tan\frac{\theta}{2} &= \frac{\Phi \pm \sqrt{-\hat{l}_{min}^2 + \Phi^2 + \Upsilon^2}}{\hat{l}_{min} + \Upsilon} \end{aligned}$

Table 6.3: **Various reduced dimensionality solutions** *The initialisation problem can be solved analytically for many subspaces of the affine space. Several W spaces are presented along with their analytic solutions. See sections 6.2.10 and 6.2.11 for a description of the different systems, their constraints and their solutions.*

One of the reasons that these methods have had such success is that they provide good ways to find two-dimensional features from one-dimensional data. Instead of trying to compare one curve with another, features are extracted from each curve, and only the features are compared. There are many such features that are used, including discontinuities in the first and second derivatives, bitangencies and inflection points. Each of these features gives point correspondences between the two curves, thus eliminating the problems relating to parametrisation of the curve and the aperture problem. In addition, redundant information can be used to confirm the identity of the object being observed, thus preventing the initialisation from being distracted by image noise.

In some ways, representing curves by B-splines simplifies the problem of computing these features. With quadratic B-splines, each span of the B-spline is a portion of a parabola. As a result, inflection points and discontinuities in the first and second derivatives can only occur at the boundaries between spans. Additionally, bitangencies must occur on non-adjacent spans (unless there are multiple knots), and there are closed-form solutions for determining when and where they occur, if at all. Another strength of the invariance method is that it can handle the case when the contours being compared are open, and/or occluded. Moment based methods are known to be sensitive to occlusions, and it is not clear how to define the moments of an open spline.

Unfortunately, the accuracy required of these invariant measures can be a limitation. Invariants have been used for model-based vision where exact models are known ahead of time, and where the deformation between the model and the observed image is very closely affine or projective. With biological objects, such as hands and plants, or when one object out of a class of objects is being tracked (such as a generic car instead of a particular make or model), the deformation between the observed example and the original model is only approximately affine. While the 'invariant' techniques still could prove to be useful for biological systems, research into this field is still in its infancy (Fleck et al., 1996; Yow and Cipolla, 1995).

6.3 Moment-based initialisation with general shape spaces

The affine space is a very convenient basis for tracking. It is a linear model, so it integrates well into the Kalman filter framework. Motions described by an affine deformation can also be described by the rigid motion of an affine camera, a system that is both well behaved and well understood. Affine deformations are equivalent to a change of basis, which means that there is a complete mapping from $I\!R^2$ to $I\!R^2$. A consequence of this is the closed-form initialisation equations described in 6.2.4 that readily allow the affine deformation to be recovered from global measurements without ever considering the actual parametrisation of the B-spline (see equation 6.20).

When a more general shape space (such as a shape space that allows rigid motion of the hand plus articulated thumb motion) is used for tracking, the problem is more complicated. It is still easy to describe how a point on the contour moves under the specified deformation, but in general there is no motion of a camera or 'global' deformation that achieves the same effect. The movements of points off the contour are undefined, and it becomes difficult to invert and/or recover the deformation.

The equations describing the relationship between the moments of area and the shape of the contour become more complicated than in the affine case and the problem of reconstructing a location within a model space from measuring moments quickly becomes intractable. The zeroth order moment (area) is quadratic in the control points (see equation A.9 in the appendix), and consequently in the model space, the first order moments (area times the centroid) are cubic (see equations A.10 and A.11), and the second order moments are quartic (see equations A.12, A.13 and A.14). The equation for the centroid turns out to be a cubic divided by a quadratic with no obvious simplification possible.

As a result of this additional complication, there are two different paths that can be used to obtain the initial estimate of the target's position. Either some approximation can be used that gives a closed-form solution, or numerical methods can be used to recover the

location of the target exactly.

6.3.1 Using the affine initialisation for general shape spaces

Many general shape spaces have all or part of the affine shape space contained within them. In this case, it may be possible to get an initialisation in the general shape space by first finding the affine deformation that best relates the template to the image. The affine deformation can then be projected into the general shape space and used as an initialisation.

Given a general shape space defined by W_{gen} and an affine shape space defined by W_{aff} with both based on the same template, it is simple to relate a point in the general state space $\mathbf{x}_{\text{gen}}$ to the nearest point in the the affine state space $\mathbf{x}_{\text{aff}}$ where 'nearest' refers to the shape of the B-spline corresponding to each state. Unlike section 6.1.1 where one of the curves had an unknown parametrisation, in this case both have known parametrisations. For each W matrix relating the location in state space to a set of control points, there exists a corresponding M matrix relating the control points to the state space. For details on constructing the M matrix, see (Blake et al., 1993a) and (Curwen, 1993).

Thus, given an initialisation in the affine space $\mathbf{x}_{\text{aff0}}$, the 'nearest' initialisation point $\mathbf{x}_{\text{gen0}}$ in the general state space is

$$\mathbf{x}_{\text{gen0}} = M_{\text{gen}} W_{\text{aff}} \mathbf{x}_{\text{aff0}}. \tag{6.49}$$

The affine solution is first projected into the control point space, then projected into the general shape space. If the affine space is a subspace of the non-affine model space, the B-spline specified by $\mathbf{x}_{\text{gen0}}$ and $\mathbf{x}_{\text{aff0}}$ are identical. An example of this has already been shown in figure 6.11. In this case, the general state space describing the motion of the lips includes all six affine degrees of freedom, so the affine initialisation is a 'legal' initialisation of the general state space.

In the case that only five of the affine degrees of freedom are contained in the general shape space, the B-splines specified by $\mathbf{x}_{\text{gen0}}$ and $\mathbf{x}_{\text{aff0}}$ will differ. However, it is possible to

use the reduced-dimensionality solution from section 6.2.10 to find the best affine initialisation under the constraint that the affine degree of freedom not in the general shape space is set to zero.

Sometimes a particular affine degree of freedom is contained in the general state space, but is known to vary little. In this case it is sometimes better to assume that only five of the affine degrees of freedom are contained in the general shape space and use the reduced dimensionality solution. Again, this is true in the case of the lips – the v_x component of the affine deformation is always near zero. Restricting the initialisation to use the five other affine degrees of freedom gives much improved results. In figure 6.5, the lips are initialised using this method; since the prior for this model is very strong, initialisation is much better than with the other methods (compare figure 6.5 with the bottom row of figure 6.11).

6.4 Numerical solutions for initialising general shape spaces

As mentioned before, the functions relating a B-spline and its moments are complex. In general, the functions are non-invertible: for any set of moments there are many corresponding B-splines. However, the problem of initialising an Active Contour with a general shape space is slightly different. A set of legal B-splines is specified ahead of time by the shape space, and the problem of initialising is finding the B-spline from that set that has the same moments as a segmented image.

There are many numerical methods for solving problems such as these. In this section, unless otherwise mentioned, we will use Broyden's method (Press et al., 1988). Broyden's method is a multidimensional generalisation of the secant method, described by Numerical Recipes (Press et al., 1988) as a globally convergent method for solving nonlinear systems of equations, which converges to a solution from almost any starting point.

6.4.1 Five d.o.f. initialisation in a general state space

To use Broyden's method, it is necessary to define a function $\mathbf{f}(\mathbf{x})$ that takes an input vector (a point $\mathbf{x}$ in the state space) and returns an output vector (the difference between moments of the B-spline corresponding to $\mathbf{x}$ and the moments measured in the image).

Because the third order moments were found to be sensitive to noise (figure 6.12), only the zeroth, first and second order moments will be used. Broyden's method, as well as most numerical root-finding algorithms (Press et al., 1988) use the Jacobian matrix to choose how to refine the current estimate. If the Jacobian is singular, the method fails. Therefore the input and output vector must be of the same dimension. In practice the zeroth moment is not sufficiently independent of the other vectors. As a result, only five degrees of freedom of the general shape space can be recovered by matching the first and second moments of the spline to those of the image.

Under these constraints, the problem of initialising the five degrees of freedom consists of finding some vector $\mathbf{x}$ such that

$$c'_x = c_x\left(W\mathbf{x} + \overline{\mathbf{X}}\right) \tag{6.50}$$

$$c'_y = c_y\left(W\mathbf{x} + \overline{\mathbf{X}}\right) \tag{6.51}$$

$$c'_{xx} = c_{xx}\left(W\mathbf{x} + \overline{\mathbf{X}}\right) \tag{6.52}$$

$$c'_{xy} = c_{xy}\left(W\mathbf{x} + \overline{\mathbf{X}}\right) \tag{6.53}$$

$$c'_{yy} = c_{yy}\left(W\mathbf{x} + \overline{\mathbf{X}}\right) \tag{6.54}$$

where $c'_{xx}, c'_{xy}, c'_{yy}, c'_x$, and c'_y are the first and second moments of the shape in the thresholded image, and $c_{xx}(\mathbf{X}), c_{xy}(\mathbf{X}), c_{yy}(\mathbf{X}), c_x(\mathbf{X})$, and $c_y(\mathbf{X})$ are now functions that return the first or second moment of the B-spline specified by the control point vector $\mathbf{X}$.

6.4.2 Separating translation from other degrees of freedom

Many times, a shape space explicitly includes independent x and y translation in the W-space. In this case, it is possible to initialise all five degrees of freedom by minimising a

three d.o.f. system. By using the mean-centred second-order moments (i.e. c_{xx}) and not the "raw" second order moments (i.e. s_{xx}), the effects of translation are separated from the other degrees of freedom. Then the problem can be solved as follows:

Assume that the W matrix has five degrees of freedom of the form

$$W = \left(\begin{pmatrix} \mathbf{1} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \end{pmatrix}, (\mathbf{q}_1), (\mathbf{q}_2), (\mathbf{q}_3) \right), \tag{6.55}$$

so that the state can be described as

$$\mathbf{X} = x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3 + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \overline{\mathbf{X}}. \tag{6.56}$$

x_1, x_2 and x_3 can then be recovered by solving the following equations:

$$c'_{xx} = c_{xx}\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3 + \overline{\mathbf{X}}\right) \tag{6.57}$$

$$c'_{xy} = c_{xy}\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3 + \overline{\mathbf{X}}\right) \tag{6.58}$$

$$c'_{yy} = c_{yy}\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3 + \overline{\mathbf{X}}\right). \tag{6.59}$$

Note that the translational component of $\mathbf{X}$ has been removed from the equations because the second order moments are invariant to translation.

Once x_1, x_2 and x_3 have been recovered, it is possible to solve the equations to obtain the translational degrees of freedom, giving the initial state of

$$\begin{pmatrix} c'_x - c_x\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3 + \overline{\mathbf{X}}\right) \\ c'_y - c_y\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3 + \overline{\mathbf{X}}\right) \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{6.60}$$

For certain shape spaces and templates, there are even simpler ways to recover the initial position. Appendix A.4 describes special cases when the position can be recovered by solving a two or even one degree of freedom system.

Figure 6.13 shows an example comparing the non-linear initialisation method from this section and initialising using the affine approximation of section 6.3.1.



optimal 5 d.o.f. affine initialisation 5 d.o.f. non-linear initialisation

Figure 6.13: **Non-linear initialisation for general shape spaces** *Two different methods of initialising non-affine shape spaces are compared. The shape-spaces used were obtained through a PCA analysis of tracking data. The PCA was truncated at eight degrees of freedom. The PCA retained all six affine degrees of freedom. By determining the five most-significant affine degrees of freedom remaining in the PCA space, the figure on the left was initialised using the methods described in sections 6.2.11 and 6.3.1. The figure on the right was initialised using the numerical methods described in section 6.4. Because the non-linear method initialises the most significant degrees of freedom, in general it gives better initial estimates. In this case, note that the corners of the mouth are better aligned and that when a normal search is performed on the right image, the search finds the correct edge all around the mouth, whereas the affine initialisation method will lock onto the wrong edge at the corners of the mouth.*

6.5 Summary

This chapter has studied the initialisation problem part by part and as a whole. A description of the difficulty of the initialisation problem was presented, including a theoretical justification of the use of normal searches as the best way to localise features on a B-spline and provided measurable criteria for when the normal search is no longer appropriate. The initialisation of the state estimate of the Kalman filter is explored in detail. Previous work for initialising the Active Contours was surveyed.

A detailed look at using moments of area as the criteria for initialisation was presented for a number of different shape spaces. For affine shape spaces, and spaces contained within the affine shape space, it is possible to robustly recover five degrees of freedom by comparing the first and second moments of area of the target in the initial image and the template of the target. To recover all six affine degrees of freedom, it is necessary to use some other information. Three methods were compared on a variety of images: comparing some function of the third moments in the template and image, choosing the solution with the greatest common area, and choosing the solution with the least displacement along the normal. The third moment is shown to be an unstable method for recovering the last degree of freedom in images with noise. The maximum common area method performs slightly better than the least displacement methods. However, the least displacement method is much faster, and is preferred in time constrained situations. Both perform much better than the third moment method.

The problem of finding the initial state estimate for a B-spline with a non-affine shape space is much more difficult than the affine case. In the affine case, there were analytic formulae relating the moments of the target and the moments of the template to the initial state estimate. The equivalent equations for the general (non-affine) case are analytically insoluble. Two approximations to the solution were presented. Because many non-affine shape spaces contain some or all of the affine shape space, in some cases it is possible to

find the best affine initialisation for an image, project that initialisation into the non-affine shape space, and use the projection as an approximate initialisation. Alternatively, it is possible to solve the initialisation problem through numerical non-linear equation solvers such as Broyden's method. Because Broyden's method can initialise the most significant degrees of freedom (i.e. the first n components of the PCA) instead of the the degrees of freedom that happen to correspond to the affine space, it provides better initialisation. However, because it is an iterative numerical method, it can fail to converge. Because its computational cost is related to the dimension of the system it is solving, some methods to reduce the dimensionality of the problem are shown.

7

Initialisation of the state uncertainty

The previous chapter addressed the issue of finding the initial state estimate for the Kalman filter of the Active Contour. However, this is only part of the information needed to initialise the filter. Before tracking begins, it is also necessary to set the initial uncertainty of the Kalman filter.

Initialising uncertainty in Kalman Filters is a subject that is often ignored or avoided. Even books such as *Kalman Filtering: Theory and Practice* by Grewal and Andrews (Grewal and Andrews, 1993) that exclusively discuss Kalman Filters devote little (if any space) to this topic. For Active Contours, initialising the uncertainty is crucial. Much of the stability of Active Contours comes from searching over the right number of pixels. The covariance matrix of the Kalman Filter determines the search scale. If too few pixels are searched (i.e. the uncertainty is underestimated), features may be missed. If too many pixels are searched (i.e. the uncertainty is overestimated), the Active Contours are more likely to be distracted by background features.

Several well-known and novel covariance initialisation algorithms are examined. Iterating the Ricatti equation and computing the prior from training sequences are shown to

be equivalent methods for initialising the covariance matrix when the training sequence is long. Additionally, several new methods are presented that initialise the covariance matrix while taking into account additional information provided by a state initialiser (such as in chapter 6).

7.1 Estimating uncertainty from learnt dynamics

The standard practice (Grewal and Andrews, 1993) for determining the initial uncertainty of a Kalman Filter is to iterate its covariance update (or Ricatti) equation

$$P_{k+1} = FP_kF^T + Q \tag{7.1}$$

until it converges. The resulting P_∞ is the prior uncertainty associated with a dynamical system, where no measurements have ever been made. Provided the dynamics are stable ($\|F\|_\infty < 1$), this covariance will converge.

Often, the system dynamics of the Active Contour have been learnt through observing a tracked sequence and then applying the learning algorithm as specified in (Blake et al., 1995). In this case, it is possible to come up with an alternative estimate for the initial covariance based on the observed sequence. Assuming a zero mean sequence, it is possible to compute the covariance of the state of the observed sequence:

$$\frac{1}{n-2}\sum_{i=1}^{n-1}\begin{pmatrix}\mathbf{X}_i\\ \mathbf{X}_{i+1}\end{pmatrix}\begin{pmatrix}\mathbf{X}_i^T & \mathbf{X}_{i+1}^T\end{pmatrix}, \tag{7.2}$$

which can be rewritten using the learning notation as

$$\begin{pmatrix}S_{00} & S_{01}\\ S_{10} & S_{11}\end{pmatrix}. \tag{7.3}$$

Because this specifies the uncertainty of the entire state, it is arguably an appropriate starting covariance for the Kalman filter. The question becomes which is the *correct* initialiser to use.

Fortunately, in the case of a long, stable training sequence, the two turn out to be exactly the same. By substituting the covariance matrix of the state of the observed sequence

into the Ricatti equation, we can see that the observed covariance matrix is, in fact, an approximate solution to the Ricatti equation:

$$\begin{aligned} & F \begin{pmatrix} S_{00} & S_{01} \\ S_{10} & S_{11} \end{pmatrix} F^T + Q \\ = & \begin{pmatrix} 0 & I \\ A_0 & A_1 \end{pmatrix} \begin{pmatrix} S_{00} & S_{01} \\ S_{10} & S_{11} \end{pmatrix} \begin{pmatrix} 0 & A_0^T \\ I & A_1^T \end{pmatrix} + Q \\ = & \begin{pmatrix} S_{11} & S_{10}A_0^T + S_{11}A_1^T \\ A_0S_{01} + A_1S_{11} & A_0(S_{00}A_0^T + S_{01}A_1^T) + A_1(S_{10}A_0^T + S_{11}A_1^T) \end{pmatrix} + Q. \end{aligned} \tag{7.4}$$

Since the dynamics A_0 and A_1 were just learnt from the same training data, the following equations are true:

$$S_{20} = A_0S_{00} + A_1S_{10} \tag{7.5}$$

$$S_{21} = A_0S_{01} + A_1S_{11}. \tag{7.6}$$

These reduce equation 7.4:

$$\begin{aligned} & \begin{pmatrix} S_{11} & S_{10}A_0^T + S_{11}A_1^T \\ A_0S_{01} + A_1S_{11} & A_0(S_{00}A_0^T + S_{01}A_1^T) + A_1(S_{10}A_0^T + S_{11}A_1^T) \end{pmatrix} + Q \\ = & \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & A_0S_{20}^T + A_1S_{21}^T \end{pmatrix} + Q \end{aligned} \tag{7.7}$$

If the data sequence is long enough

$$S_{ab} \approx S_{a+1,b+1} \quad \forall a, b. \tag{7.8}$$

Additionally, Q is of the form

$$\begin{pmatrix} 0 & 0 \\ 0 & C \end{pmatrix} \tag{7.9}$$

where C has been computed by the learning algorithm to be $S_{22} - A_0S_{02} - A_1S_{12}$. This

further reduces equation 7.7:

$$\begin{aligned}
& \begin{pmatrix} S_{11} & S_{21}^T \\ S_{21} & A_0 S_{20}^T + A_1 S_{21}^T \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & C \end{pmatrix} \\
= & \begin{pmatrix} S_{00} & S_{10}^T \\ S_{10} & A_0 S_{20}^T + A_1 S_{21}^T \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & S_{22} - A_0 S_{02} - A_1 S_{12} \end{pmatrix} && (7.10) \\
= & \begin{pmatrix} S_{00} & S_{10}^T \\ S_{10} & A_0 S_{20}^T + A_1 S_{21}^T + S_{22} - A_0 S_{02} - A_1 S_{12} \end{pmatrix} && (7.11) \\
= & \begin{pmatrix} S_{00} & S_{10}^T \\ S_{10} & S_{22} \end{pmatrix} && (7.12) \\
= & \begin{pmatrix} S_{00} & S_{01} \\ S_{10} & S_{11} \end{pmatrix}. && (7.13)
\end{aligned}$$

Therefore the covariance matrix of the observed sequence is an approximate solution to the Ricatti equation for long data sequences.

7.1.1 Estimating initial uncertainty from training sequences

The previous section described two methods for estimating the initial uncertainty of the Kalman filter from data collected from training sequences. Implicit in both methods was the assumption that the prior of $\mathbf{x}_0$ is the same as the prior for the state at any other time. There are many times when the prior for $\mathbf{x}_0$ is significantly different than the rest of the states $\mathbf{x}_i$ (i.e. the states corresponding to the ith image of the data sequence). In the application of tracking cars on a motorway using overhead cameras, Ferrier, *et. al.* (Ferrier et al., 1994) used a motion alarm mechanism (see section 6.2.2) that only started tracking when a car passed through a small, predetermined section of the image. In the case of tracking plants from a tractor-mounted camera, it can accurately be predicted where new plants are likely to appear in the image (Marchant and Brivot, 1995). Both of these systems have special initial conditions that are more restrictive than the priors estimated by section 7.1. It would be desirable to be able to learn these special initial conditions through training sequences in the same way that training sequences have been used to learn dynamical models.

One problem with learning initial conditions is that it is difficult to collect enough data to get a good estimate on the prior of $\mathbf{x}_0$. When learning dynamical models, each training

sequence normally contributes thousands of data vectors towards the learning. Even when 'short' training sequences are used, such as those described in section 5.4, each sequence typically contributed over 100 six dimensional data points towards learning a 93 dimensional system (36 d.o.f.'sfor A_0, 36 d.o.f.'sfor A_1, and 21 d.o.f.'sfor B). By combining several training sequences it was possible to get a good estimate of the underlying system. In the case of learning the initial conditions of an affine system, each training set contributes one 12 dimensional vector (a $\mathbf{x}_0, \mathbf{x}_1$ pair – one for position, one for velocity) towards learning a symmetric 12×12 covariance matrix. Thus a minimum of thirteen sequences are needed to get a covariance matrix of full rank, and two to three times as many are desirable to get a good estimate of the covariance.

A second problem with learning the initial covariance is getting reliable estimates for $\mathbf{x}_0$ and $\mathbf{x}_1$ from training data. Training data is typically gathered from the Active Contour tracker (Blake et al., 1995; Ferrier et al., 1994; Moses et al., 1995). However, the Active Contour tracker usually takes a few timesteps to 'lock on' to the target. Until it is 'locked', the estimates of the state are unreliable.

There are two different ways in which the estimates for $\mathbf{x}_0$ and $\mathbf{x}_1$ can be improved. For convenience label the Γ different sequences of images to be used for training I_i^γ, where $\gamma \in 1 \ldots \Gamma$, $i \in 1 \ldots n^\gamma$ and I_i^γ corresponds to the ith image of the γth sequence. The training sequences will be labelled as in chapter 5, so $\mathbf{x}_i^\gamma$ is the data point in the γth training sequence that corresponds to the estimated state of the tracker after it has processed image I_i^γ.

It is possible to improve the estimate of $\mathbf{x}_0^\gamma$ by letting the tracker iterate on image I_0^γ until it converges. Similarly, running the tracker until convergence on image I_1^γ will improve the estimate of $\mathbf{x}_1^\gamma$. This set of initial estimates $\mathbf{x}_0^1, \ldots, \mathbf{x}_0^\Gamma, \mathbf{x}_1^1, \ldots, \mathbf{x}_1^\Gamma$ can then be used to improve the initial state and uncertainty estimates of the Kalman filter.

A second possible method to improve the estimates of the initial state is to use a Rauch-Tung-Striebel smoother (Grewal and Andrews, 1993). Kalman filters are optimal in the sense that they provide the best estimate of the state and uncertainty at any time n given

the current measurements and all past measurements. The past measurements provide temporal continuity that helps reduce the effects of measurement noise on the estimate. The reason that it is difficult to estimate the state and uncertainty at time 0 and 1 is that there are no previous measurements and thus no temporal continuity.

The Rauch-Tung-Striebel smoother is a derivative of the Kalman filter that provides the best estimate of $\mathbf{x}_i$ given the current measurements, all past measurements, and all future measurements. It does this by examining an entire sequence at once, taking the outputs of the Kalman filter and running it backwards. The equations to do this are as follows:

$$\hat{\mathbf{x}}_{[s]n} = \hat{\mathbf{x}}_{n|n} \tag{7.14}$$

$$\hat{\mathbf{x}}_{[s]k} = \hat{\mathbf{x}}_{k|k} + A_k(\hat{\mathbf{x}}_{[s]k+1} - \hat{\mathbf{x}}_{k+1|k}) \tag{7.15}$$

$$A_k = P_{k|k}F^T P^{-1}_{k+1|k} \tag{7.16}$$

$$P_{[s]k} = P_{k|k} + A_k(P_{[s]k+1} - P_{k+1|k})A_k^T \tag{7.17}$$

where $\hat{\mathbf{x}}_{k|k}$ and $P_{k|k}$ are the Kalman filter's estimate of the state at time k and its uncertainty, $\hat{\mathbf{x}}_{k+1|k}$ and $P_{k+1|k}$ are the Kalman filter's prediction of the state at time $k+1$ and its uncertainty, $\hat{\mathbf{x}}_{[s]k}$ is the Rauch-Tung-Streibel smoothed state estimate at time k, and $P_{[s]k}$ is the covariance of the uncertainty of the smoothed estimate.

By applying Rauch-Tung-Striebel to all Γ training sequences, improved estimates $\hat{\mathbf{x}}^1_{[s]0}, \ldots \hat{\mathbf{x}}^\Gamma_{[s]0}$ and $\hat{\mathbf{x}}^1_{[s]1}, \ldots \hat{\mathbf{x}}^\Gamma_{[s]1}$ with their uncertainties are obtained. These can be combined using least-squares estimation weighted by the $P^i_{[s]k}$ terms to obtain the desired initial state estimate and uncertainty.

It is important to note that the $P_0 \neq P_{[s]0}$. $P_{[s]0}$ is the uncertainty of $\hat{\mathbf{x}}_{[s]0}$ given the entire sequence, whereas P_0 describes the uncertainty of the state without any knowledge of measurements yet to come.

7.1.2 Estimating initial uncertainty for an initialiser

Once an initialiser has been used to find $\hat{\mathbf{x}}_0$, computing the initial state uncertainty is actually a similar problem to that of Ferrier *et. al.* (1994): there is more information about the initial position $\mathbf{x}_0$ than there is about an arbitrary point $\hat{\mathbf{x}}_i$ later in the sequence. In particular, it would be hoped that using an initialiser would greatly reduce the initial uncertainty so that $\|P_0\| \ll \|P_\infty\|$. The initial uncertainties as defined by equations 7.1 and 7.2 will overestimate the uncertainty.

Given a set of initial state estimates $\hat{\mathbf{x}}_0^1, \ldots \hat{\mathbf{x}}_0^\Gamma$ obtained through an initialisation method, and smoothed initial state estimates $\hat{\mathbf{x}}_{[s]0}^1, \ldots \hat{\mathbf{x}}_{[s]0}^\Gamma$ from the Rauch-Tung-Striebel filter, the initial uncertainty can be approximated by

$$P_0 = \frac{1}{\Gamma - 1} \sum_{i=0}^{\Gamma} \begin{pmatrix} \hat{\mathbf{x}}_{[s]0}^i - \hat{\mathbf{x}}_0^i \\ \hat{\mathbf{x}}_{[s]1}^i - \hat{\mathbf{x}}_1^i \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}_{[s]0}^i - \hat{\mathbf{x}}_0^i \\ \hat{\mathbf{x}}_{[s]1}^i - \hat{\mathbf{x}}_1^i \end{pmatrix}^T . \tag{7.18}$$

This is an approximation because $\hat{\mathbf{x}}_{[s]0} \neq \mathbf{x}_0$.

7.2 Combining initialisation of state estimate and initialisation of uncertainty

Until this point, the initialisation problem has been considered as being completely distinct from the tracker itself. Because moments such as the ones used for the initialisation are actually measurements made on the image, it might seem to be a good idea to incorporate these measurements within the existing Kalman filter framework. While it is possible to do so, the filter must be modified to handle the changes.

The first complication arises from the Kalman filter being a linear filter. The state and state uncertainty at one timestep are a linear function of the state and state uncertainty at previous timesteps. The measurements are modelled as a linear function of the state and measurement noise. A measurement $\mathbf{z}_k$ is related to the state $\mathbf{x}_k$ by a matrix H in the following way:

$$\mathbf{z}_k = H\mathbf{x}_k + \nu_k \tag{7.19}$$

where ν_k is Gaussian additive noise. Because geometric moments are not linear functions of the states, the relationship between the state and measurements is no longer linear. Instead, the measurement $\mathbf{z}_k$ is now a general function of the state and noise:

$$\mathbf{z}_k = H(\mathbf{x}_k, \nu_k) \tag{7.20}$$

(i.e. the noise is non-Gaussian). To simplify things slightly, it will be assumed that the noise is additive Gaussian, giving

$$\mathbf{z}_k = H(\mathbf{x}_k) + \nu_k. \tag{7.21}$$

The Kalman filter can be modified to handle non-linear measurement models (such as equation 7.21), as well as non-linear prediction models. Known as the Extended Kalman filter (or EKF), it is no longer an optimal filter, but it is still the best linear filter (Grewal and Andrews, 1993). The EKF works by linearising the Kalman filter about the current state estimate. While it is not optimal and not provably stable, it is still extensively used in practice (Grewal and Andrews, 1993; Koller et al., 1994). The measurement update equations of the EKF are:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} - K_k(\mathbf{z}_k - H(\hat{\mathbf{x}}_{k|k-1})) \tag{7.22}$$

$$K_k = P_{k|k-1}H_k^T(H_k P_{k|k-1} H_k^T + R_k)^{-1} \tag{7.23}$$

$$P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1} \tag{7.24}$$

$$H_k \triangleq \frac{\partial H}{\partial \mathbf{x}}\left(\hat{\mathbf{x}}_{k|k}\right). \tag{7.25}$$

At each timestep, these equations are linearised at the point $\hat{\mathbf{x}}_{k|k}$. The quality of the linearisation depends on how close $\hat{\mathbf{x}}_{k|k}$ is to $\mathbf{x}_{k|k}$. Before initialisation, $\mathbf{x}_0$ is undefined. Therefore H must be linearised about the system mean. This will lead to large errors in the initial state estimate.

Alternatively, the EKF could be used only to initialise the state uncertainty. A global equation solver such as Broyden's method could be used to initialise the five most significant

degrees of freedom in the state estimate $\mathbf{x}_0$ by changing the shape of the B-spline so it had the same first and second moments as were measured in the image. Equation 7.24 could then be used to estimate the uncertainty. It would be hoped that components in $\mathbf{x}_0$ that had been forced to be zero during the nonlinear minimisation would be given high uncertainty. All that remains is to estimate R, the measurement noise.

The problem of computing R is similar to the problem of estimating the initial uncertainty for an initialisation method (section 7.1.2). Before, the state uncertainty was computed directly. With the EKF, R is the uncertainty of the measurements, (i.e. the error in measuring the first and second moments in the image). To estimate R, do the following for each training sequence: (1) measure the first and second moments in the first image, storing the results in the vector $\mathbf{z}_0^\gamma$, (2) take the smoothed initial state estimates $\hat{\mathbf{x}}_{[s]1}^\gamma$ and find the moments of that estimate $H(\hat{\mathbf{x}}_{[s]1}^\gamma)$. R can then be evaluated

$$\begin{aligned} R &\triangleq \mathsf{E}\left((\mathbf{z} - H(\mathbf{x}))\,(\mathbf{z} - H(\mathbf{x}))^T \right) \\ &= \frac{1}{\Gamma - 1} \sum_{i=0}^{\Gamma} \left(\mathbf{z}_0^\gamma - H(\hat{\mathbf{x}}_{[s]1}^\gamma) \right) \left(\mathbf{z}_0^\gamma - H(\hat{\mathbf{x}}_{[s]1}^\gamma) \right)^T . \end{aligned} \tag{7.26}$$

In the end it is difficult to see what has been gained. By directly estimating the initial uncertainty from training sequences, such as in section 7.1.2, a value of P_0 has been estimated directly from training sequences. In this section, a measurement noise R has been learnt after making several assumptions about the noise being Gaussian and the H function being close to its first order Taylor expansion. This linearisation of H is used several times, inverted and eventually gives a value of P_0. The argument for simplicity is convincing.

7.3 Summary

The problem of estimating the uncertainty of the initial state was discussed. Previously, the effects of initialisation routines on the uncertainty were ignored. Three methods were commonly used: the initial uncertainty was hand-tuned P_∞ was used, or the observed variance in a training sequence was used. The latter two methods were shown to be equivalent

in the case of learnt dynamics from a large training sequence.

When the state has been automatically initialised through any method, the uncertainty of the initial estimate will be less than P_∞ because P_∞ is the uncertainty of the system in the absence of measurements. The measurement provided by the state initialisation will necessarily lower this uncertainty. Even when no initialiser has been used, there may be other information that reduces this uncertainty, such as the knowledge that tracking targets are more likely to enter in a specific region of the image.

To avoid misestimating the initial uncertainty, it is possible to learn the initial uncertainty from observed data sequences in a similar way that it was possible to learn the model from observed data sequences. Several methods were presented to learn the initial uncertainty from observed data sequences both when initialisers are used and when they aren't. An additional method is presented that uses an Extended Kalman Filter to simultaneously initialise the state and uncertainty of the Kalman filter.

8

Conclusion and future work

The research described in this thesis describes work towards making Active Contours fully autonomous mechanisms capable of tracking biological targets in real-world situations. The problem of tracking the apparent motion of cauliflower plants photographed from a tractor in a field was of special interest. In order to reach this simple goal it was necessary to complete research in a number of different areas.

The introductory chapter briefly stated the problems that were discussed in the thesis. The literature survey in the second chapter provided a discussion of previous works by various authors in related fields.

Chapter three explored the theory of classification as it relates to classifying colour images. First, the theory of Bayesian classification was discussed, and the notion of finding the upper limit of accuracy was introduced. Establishing the upper limit of accuracy of a classifier is useful because it allows methods of classification to be rejected on the basis of a few training sets. The method was used in chapter four to show that in black and white images of cauliflower fields it was impossible to reliably classify individual pixels as to whether they were plant or soil.

Colour images were examined as a logical alternative to black and white images. There are many different ways to represent colour; several of these different colour representations were examined. The dichromatic reflection model (DRM) was introduced as a way to predict the perceived colour of the different data objects in an image. The DRM models the perceived colour of an object as a linear combination of the colour of the "body" of the object and the colour of the light. The relationship between the DRM and colour-spaces was also explored. Specifically, it was noted that in the case that when white light illuminates an object of uniform body colour, the perceived hue of that object is constant and independent of the amount of illumination.

Chapter four applied the theory of the previous chapter to the problem of classifying pixels from images of cauliflower fields as being soil, cauliflower or weeds. It was shown that black and white images provide insufficient data for pixel-by-pixel classification. However, classification was possible in colour images of the same fields. A linear classifier was provided for classification in RGB space and a simple classifier was also provided for classification in hue space (i.e. HSV space ignoring the saturation and value of the pixels).

Once it was possible to identify individual pixels as pertaining to a specific class, the next step was to get the Active Contour tracker working on the data. However, early results showed poor tracking, exposing several shortcomings of the previous learning algorithms for Active Contours. The learning sequences of the data were very short (under fifty frames); in this case it was discovered the mean of the underlying system was not the mean of the observed sequence of data. Secondly, the previous learning algorithm assumed that all sequences had the same mean. The cauliflower data showed very different means from one sequence to another. The learning algorithm was reformulated to compensate for both of these cases.

The next problem had to do with the difference between tracking an object and a class of objects. In any data sequence, a single cauliflower was tracked. During the sequence the plant did little except translate. However, the change from one sequence to another was

great since a different plant was being tracked in the second sequence. In the previous model these two sources of variance were lumped together resulting in a system that allowed too much freedom in the tracking of an individual plant, and not enough freedom when adjusting from one plant to another. To solve this problem, a new system model was proposed that modelled these two sources of variance separately. The augmented model was tested both on completely simulated data, and on real images that were altered to add distracting features. Its performance was shown to be superior to the older model.

Once the Active Contour mechanism was working well, the remaining piece of the puzzle was the problem of automatically initialising the Active Contour. Chapters six and seven discussed the initialisation problem in detail. The general problem of fitting a B-spline to a target in an image was discussed, as well as the problem of fitting a B-spline to a target when the B-spline was constrained to lie in a shape space (i.e. finding the initial state estimate of the Kalman filter). The solutions to the latter problem depend greatly on which shape space the B-spline lies in. For shape spaces contained in the affine space, three solutions were presented: one analytic solution and several novel solutions that are part-analytic, part-numeric solutions and are robust in the presence of noise. For more general shape spaces, several novel techniques were discussed for initialising some of the degrees of freedom of the system.

To initialise the rest of the Kalman filter, it is necessary to find the uncertainty of the initial state estimate. Chapter seven shows previous methods to be inappropriate in several cases, including the case when an initialiser is used to find the initial state estimate. Several novel techniques were discussed for determining the uncertainty by learning from training sequences, and also for determining the uncertainty of a system when an initialiser is used.

This thesis chronicles several advances in both the theory and practice of visual tracking. However, as with any academic work, there are as many new areas of research that open up as there are that close. The following sections detail a few of them.

8.1 Automatic redimensioning of the Kalman filter

Some realistic tracking systems (for example, the mean estimation system detailed in chapter five) have dynamics that are uncontrollable. Uncontrollability of a Kalman filter's dynamics means that the covariance of at least one the dimensions of the states of the filter approaches zero as time goes towards infinity. A zero covariance doesn't affect the tracking performance of the filter *per se* – a zero covariance merely states that a given dimension of the state vector is known exactly. However, at each timestep the filter still must go through significant computations to reconfirm what is already known: that the state hasn't changed. The time is wasted, and in realtime systems, there is never time to waste.

Instead of this unnecessary computation, a much more desirable system would be one that detects if the system's covariance matrix has become singular, and then reformulates itself so that it operates in a lower dimension. This would reduce the computational load of the system, allowing it to do more useful things, such as track more objects or drop less frames.

The argument can be made that this reduction in work doesn't matter – the hardware must still be able to track in the worst case, i.e. when the original (full) system still has a non-singular covariance matrix. There are several reasons why this isn't true. If the system has to run at slightly less than frame rate for a few frames while the system is full but can later change to a lower dimensional system, it will perform better than an equivalent system that always runs at less than frame rate. Secondly, when multiple, independent objects are being tracked, the initialisation period of one object does not necessarily coincide with the initialisation period of the other objects in view. So a system could realistically track two objects of reduced dimensionality while tracking one object of higher dimensionality.

To incorporate this new idea, the state equations must again be modified only slightly.

Currently, the state equations of the mean estimation model are

$$\mathbf{X}_{n+2} = A_0\mathbf{X}_n + A_1\mathbf{X}_{n+1} - \Psi\overline{\mathbf{X}}_n \tag{8.1}$$

$$\Psi\overline{\mathbf{X}}_{n+1} = \Psi\overline{\mathbf{X}}_n. \tag{8.2}$$

As the covariances relating to $\overline{\mathbf{X}}_n$ diminishes to the point where the system reduces the dimension, the information in $\overline{\mathbf{X}}_n$ becomes a constant, and the equations are quite elegantly reduced to

$$\mathbf{X}_{n+2} = A_0\mathbf{X}_n + A_1\mathbf{X}_{n+1} - \Psi\overline{\mathbf{X}}_n. \tag{8.3}$$

where $\overline{\mathbf{X}}_n$ is now a constant instead. Note that if the different directions of the covariance of $\overline{\mathbf{X}}_n$ converge at different rates, it is possible to break up $\overline{\mathbf{X}}_n$ so that it is constant in some directions and variable in others. There are also lots of other cases when this would be a useful thing to do, such as simultaneously tracking multiple cauliflower plants. In this case, not only is $\overline{\mathbf{X}}_n$ converging, the cauliflowers are attached to the ground plane, so as tracking progresses knowledge of the spacing between the plants becomes known to the point that the previously independent splines are able to be combined into a single spline consisting of multiple closed contours that has a six dimensional state space.

The key to reducing the dimensionality of the Kalman filter is developing a justifiable criterion for when it is appropriate to reduce the dimensionality of the filter. One possibility is to use the condition number of the covariance matrix. When the condition number approaches the precision of the floating point numbers being used, the matrix is essentially singular – the rounding error is contributing as much to the state estimation as the covariance itself. In this case it is possible to reduce the dimensionality of the filter without any loss of information.

However, it is theoretically possible for one direction of the covariance matrix to drop to a very low value in comparison to the other directions, only to have that direction become (relatively) more important at a later time. If the filter were reduced on the previous criteria alone, information could potentially be lost.

Fortunately, in the mean estimation case there is another criterion to assure that the dimensions will never return to prominence. The state noise for $\overline{\mathbf{X}}_n$ is identically zero and the state noise for $\mathbf{X}_n$ is not. This means that the covariance for $\overline{\mathbf{X}}_n$ is a non-increasing function and that the covariance for $\mathbf{X}_n$ will reach a steady state minimum. Therefore, when the covariance for $\overline{\mathbf{X}}_n$ becomes negligible (i.e. less than the rounding error) with respect to the state noise for $\mathbf{X}_n$ it will always remain negligible. Also, by using this second criterion, the computational complexity of the problem is significantly less than for the former (since the latter essentially assumes the covariance matrix to be block diagonal), thus simplifying the computation of the condition number.

8.2 Initialisation of the Active Contour in the absence of segmentation

Chapters six and seven developed the theory and a lot of the practice of initialisation of the Active Contour mechanism. One assumption underlying the entire system was that it was possible to segment the image even if the segmentation was only approximate. However, the question has to be asked: if it's possible to segment the image, why is it necessary to track at all?

There are several good answers to that question. Firstly, the goal of Active Contours is real-time tracking on standard hardware. The kinds of algorithms used to segment the images in many cases are too computationally expensive to run in real-time. Secondly, just because an algorithm can segment the first image in a sequence doesn't necessarily mean that it can segment subsequent images. Chapter five showed several data sets where segmentation was not possible in all parts of the image. In these cases a tracker relying solely on good segmentation will fail.

However, it is still frustrating that the initialisation methods of chapter six rely on pre-segmented images, and that the features used to initialise the contours (the area-based segmentation of the image) are different from the features used when the tracker is running

(the detection of edges in the image). A new method that would enable the initialisation of the Active Contour using the same features used to track would be a great benefit.

An alternative to segmenting the image ahead of time is to try to create an enclosed area suitable for moment-based methods by joining edges in an image. It is straightforward to find the edgels in an image, and fairly straightforward to assemble the edgels into edges. Also, once there is an area that can be measured with moments, it is relatively straightforward to initialise the Active Contour. The problem is getting from the edge map to the enclosed area. It may be possible to sample groups of edges in a manner similar to the RANSAC algorithm: choose a group of edges, close the area created by them, initialise using the method of preference, evaluate the fit, and repeat until a criteria is met (usually goodness of fit or number of iterations).

A

Additional mathematical derivations and formulae

This appendix contains several formulae and derivations that were considered too unwieldy to include in the main body of the text.

A.1 Kalman filter state equations

The heart of the Active Contour is the Kalman filter which is used to incorporate novel information (obtained through measurements) into existing information (about the state at previous times and predictions about the possible motions of the system). The equations for the Kalman filter are fairly well known, though the notation varies greatly from source to source. The equations for the filter, written using the variable names used in the rest of the thesis, are the model:

$$\mathbf{x}_k = F_{k-1}\mathbf{x}_{k-1} + \mathbf{b}_{k-1} \tag{A.1}$$

$$\mathbf{z}_k = H_k\mathbf{x}_k + \mathbf{r}_{k-1} \tag{A.2}$$

and the update equations:

$$\hat{\mathbf{x}}_{k|k-1} = F_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \tag{A.3}$$

$$P_{k|k-1} = F_{k-1}P_{k-1}F_{k-1}^T + B_{k-1} \tag{A.4}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} - K_k(\mathbf{z}_k - H_k\hat{\mathbf{x}}_{k|k-1}) \tag{A.5}$$

$$K_k = P_{k|k-1}H_k^T(H_kP_{k|k-1}H_k^T + R_k)^{-1} \tag{A.6}$$

$$P_{k|k} = P_{k|k-1} - K_kH_kP_{k|k-1}. \tag{A.7}$$

The variables have the following meanings: $\mathbf{x}_k$ is the true state at time k which is always unknown. Its transformation from one timestep to another is specified by F_{k-1} which in most cases is a constant, and $\mathbf{b}_{k-1}$ which is white Gaussian noise with $\mathsf{E}\left(\mathbf{b}_{k-1}\mathbf{b}_{k-1}^T\right) = B_{k-1}$. In most Active Contour applications B_{k-1} is constant.

$\mathbf{z}_k$ is the measurement vector at time k. It is related to the state $\mathbf{x}_k$ by the matrix H_k and the measurement noise $\mathbf{r}_{k-1}$ which is Gaussian and has covariance $\mathsf{E}\left(\mathbf{r}_{k-1}\mathbf{r}_{k-1}^T\right) = R_{k-1}$. In most Active Contour applications R_{k-1} is constant.

At each timestep, the Kalman filter determines the best state estimate $\hat{\mathbf{x}}_{k-1|k-1}$ given all previous measurements up to and including step $k-1$, and the uncertainty of that estimate P_{k-1}. This estimate and uncertainty are then updated for the current step to obtain $\hat{\mathbf{x}}_{k|k-1}$ and $P_{k|k-1}$. The measurements for the current timestep are incorporated weighted by K_k, the Kalman gain, to obtain the final estimate $\hat{\mathbf{x}}_{k|k}$ and covariance $P_{k|k}$ for the timestep.

A.2 Formulae for moments of B-splines

Chapter six refers to the complex equations relating a set of B-spline control points to the moments of area of the region they enclose. Here are those equations. They were obtained by applying Green's rule to the B-spline equations, and simplifying with the aid of Mathematica. All the following are based on the assumption that the B-splines are quadratic, closed and have no multiple knots:

$$s_0 = \frac{1}{24} \sum (-10X_{1+i}Y_i - X_{2+i}Y_i + 10X_iY_{1+i} + X_iY_{2+i}) \tag{A.8}$$

$$\begin{aligned} s_x = \frac{1}{240} \sum\ & (31X_iX_{1+i}Y_i + 31X_{1+i}^2Y_i + X_iX_{2+i}Y_i + 7X_{1+i}X_{2+i}Y_i + X_{2+i}^2Y_i \\ & -31X_i^2Y_{1+i} - 31X_iX_{1+i}Y_{1+i} - X_i^2Y_{2+i} - 7X_iX_{1+i}Y_{2+i} \\ & -X_iX_{2+i}Y_{2+i}) \end{aligned} \tag{A.9}$$

$$\begin{aligned} s_y = \frac{1}{240} \sum\ & (31X_{1+i}Y_i^2 + X_{2+i}Y_i^2 - 31X_iY_iY_{1+i} + 31X_{1+i}Y_iY_{1+i} + 7X_{2+i}Y_iY_{1+i} \\ & -31X_iY_{1+i}^2 - X_iY_iY_{2+i} + X_{2+i}Y_iY_{2+i} - 7X_iY_{1+i}Y_{2+i} - X_iY_{2+i}^2) \end{aligned} \tag{A.10}$$

$$\begin{aligned} s_{xx} = \frac{1}{6720} \sum\ & (-374X_i^2X_{1+i}Y_i - 474X_iX_{1+i}^2Y_i - 374X_{1+i}^3Y_i - 5X_i^2X_{2+i}Y_i \\ & -40X_iX_{1+i}X_{2+i}Y_i - 138X_{1+i}^2X_{2+i}Y_i - 3X_iX_{2+i}^2Y_i - 38X_{1+i}X_{2+i}^2Y_i \\ & -5X_{2+i}^3Y_i + 374X_i^3Y_{1+i} + 474X_i^2X_{1+i}Y_{1+i} + 374X_iX_{1+i}^2Y_{1+i} \\ & +2X_i^2X_{2+i}Y_{1+i} - 2X_iX_{2+i}^2Y_{1+i} + 5X_i^3Y_{2+i} + 38X_i^2X_{1+i}Y_{2+i} \\ & +138X_iX_{1+i}^2Y_{2+i} + 3X_i^2X_{2+i}Y_{2+i} + 40X_iX_{1+i}X_{2+i}Y_{2+i} \\ & +5X_iX_{2+i}^2Y_{2+i}) \end{aligned} \tag{A.11}$$

$$\begin{aligned} s_{xy} = \frac{1}{13440} \sum\ & (748X_iX_{1+i}Y_i^2 + 474X_{1+i}^2Y_i^2 + 10X_iX_{2+i}Y_i^2 + 40X_{1+i}X_{2+i}Y_i^2 \\ & +3X_{2+i}^2Y_i^2 - 748X_i^2Y_iY_{1+i} + 748X_{1+i}^2Y_iY_{1+i} + 36X_iX_{2+i}Y_iY_{1+i} \\ & +276X_{1+i}X_{2+i}Y_iY_{1+i} + 40X_{2+i}^2Y_iY_{1+i} - 474X_i^2Y_{1+i}^2 - 748X_iX_{1+i}Y_{1+i}^2 \\ & -10X_i^2Y_iY_{2+i} - 36X_iX_{1+i}Y_iY_{2+i} + 36X_{1+i}X_{2+i}Y_iY_{2+i} + 10X_{2+i}^2Y_iY_{2+i} \\ & -40X_i^2Y_{1+i}Y_{2+i} - 276X_iX_{1+i}Y_{1+i}Y_{2+i} - 36X_iX_{2+i}Y_{1+i}Y_{2+i} - 3X_i^2Y_{2+i}^2 \\ & -40X_iX_{1+i}Y_{2+i}^2 - 10X_iX_{2+i}Y_{2+i}^2) \end{aligned} \tag{A.12}$$

$$\begin{aligned} s_{yy} = \frac{1}{6720} \sum\ & (-374X_{1+i}Y_i^3 - 5X_{2+i}Y_i^3 + 374X_iY_i^2Y_{1+i} - 474X_{1+i}Y_i^2Y_{1+i} \\ & -38X_{2+i}Y_i^2Y_{1+i} + 474X_iY_iY_{1+i}^2 - 374X_{1+i}Y_iY_{1+i}^2 - 138X_{2+i}Y_iY_{1+i}^2 \\ & +374X_iY_{1+i}^3 + 5X_iY_i^2Y_{2+i} - 2X_{1+i}Y_i^2Y_{2+i} - 3X_{2+i}Y_i^2Y_{2+i} \\ & +40X_iY_iY_{1+i}Y_{2+i} - 40X_{2+i}Y_iY_{1+i}Y_{2+i} + 138X_iY_{1+i}^2Y_{2+i} + 3X_iY_iY_{2+i}^2 \\ & +2X_{1+i}Y_iY_{2+i}^2 - 5X_{2+i}Y_iY_{2+i}^2 + 38X_iY_{1+i}Y_{2+i}^2 + 5X_iY_{2+i}^3) \end{aligned} \tag{A.13}$$

As noted in the text, the equation for area is quadratic, the first order moments are cubic and the second order moments are quartic. All the equations are homogeneous. The formulae for the centred and weighted moments c_x, c_{xx} and so on, do not simplify in any obvious way.

A.3 Some lemmas about matrix and vector differentiation

Some of the results in chapter five require the use of the derivatives of matrices. This section proves some of the lemmas used in the derivations in that chapter.

In general, taking the derivative of one matrix with respect to another is an unwieldy process, because the result is a fourth order tensor. This can be seen by examining the derivative

$$\frac{\partial A}{\partial B} \tag{A.14}$$

where A is an m by m matrix and B is an n by n matrix. Taking the derivative with respect to B is actually computing n^2 different derivatives. These n^2 different derivatives in turn must be computed for each of the m^2 elements in B.

One case in which it is possible to easily take matrix derivatives is taking the derivatives of the trace of a set of matrices. There are several nice results in this case which are derived here. For simplicity, Einstein's summation convention will be used, which means that

$$(a)_{ij} = l_{ik} r_{kj} = \sum_k l_{ik} r_{kj}, \tag{A.15}$$

i.e. that a repeated subscript is one that is meant to be summed over.

The first result is the first matrix derivative of the trace of a linear function of A:

$$\begin{aligned} \frac{\partial}{\partial A_{ij}} \mathrm{tr}\,(LAR) &= \frac{\partial}{\partial A_{ij}} (L_{kl} A_{lm} R_{mk}) && \text{(A.16)} \\ &= \delta_{il}\delta_{jm} L_{kl} R_{mk} && \text{(A.17)} \\ &= L_{ki} R_{jk} = L^T_{ik} R^T_{kj} && \text{(A.18)} \\ \frac{\partial}{\partial A} \mathrm{tr}\,(LAR) &= L^T R^T. && \text{(A.19)} \end{aligned}$$

The first matrix derivative of the trace of a quadratic function of A is a similar result:

$$\begin{aligned}
\frac{\partial}{\partial A_{ij}}\mathrm{tr}\left(LAMA^TR\right) &= \frac{\partial}{\partial A_{ij}}(L_{kl}A_{lm}M_{mn}A^T_{np}R_{pk}) && \text{(A.20)}\\
&= \delta_{il}\delta_{jm}L_{kl}M_{mn}A_{pn}R_{pk} + \delta_{ip}\delta_{jn}L_{kl}A_{lm}M_{mn}R_{pk} && \text{(A.21)}\\
&= L_{ki}M_{jn}A_{pn}R_{pk} + L_{kl}A_{lm}M_{mj}R_{ik} && \text{(A.22)}\\
&= L^T_{ki}R^T_{kp}A_{pn}M^T_{nj} + R_{ik}L_{kl}A_{lm}M_{mj} && \text{(A.23)}\\
\frac{\partial}{\partial A}\mathrm{tr}\left(LAMA^TR\right) &= L^TR^TAM^T + RLAM && \text{(A.24)}
\end{aligned}$$

In the case that the derivative of a matrix is taken with respect to a vector, there are a similar set of results. For a the trace of linear and quadratic functions of a matrix:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{x}_i}\mathrm{tr}\left(\mathbf{a}\mathbf{x}^T\right) &= \frac{\partial}{\partial \mathbf{x}_i}\mathrm{tr}\left(\mathbf{x}\mathbf{a}^T\right)\\
&= \frac{\partial}{\partial \mathbf{x}_i}\mathbf{a}_j\mathbf{x}_j\\
&= \delta_{ij}\mathbf{a}_j\mathbf{x}_j\\
&= \mathbf{a}_i\mathbf{x}_i\\
\frac{\partial}{\partial \mathbf{x}}\mathrm{tr}\left(\mathbf{a}\mathbf{x}^T\right) &= \mathbf{a}, && \text{(A.25)}\\
\frac{\partial}{\partial \mathbf{x}_i}\mathrm{tr}\left(\mathbf{x}\mathbf{x}^T\right) &= \frac{\partial}{\partial \mathbf{x}_i}\mathbf{x}_j\mathbf{x}_j\\
&= \delta_{ij}\mathbf{x}_j + \delta_{ij}\mathbf{x}_j\\
&= \mathbf{x}_i + \mathbf{x}_i\\
\frac{\partial}{\partial \mathbf{x}}\mathrm{tr}\left(\mathbf{x}\mathbf{x}^T\right) &= 2\mathbf{x}. && \text{(A.26)}
\end{aligned}$$

A.4 Using homogeneous equations to simplify initialisation

When initialising Active Contours with certain shape spaces and templates, it is possible to reduce the system of three simultaneous non-linear equations to a system of two or even one simultaneous non-linear equations. As can be seen by equations A.12, A.13 and A.14, c_{xx}, c_{xy} and c_{yy} are homogeneous second order functions. This means that for any vector

$\mathbf{X}$ and scale factor a

$$c_{xx}(a\mathbf{X}) = a^2 c_{xx}(\mathbf{X}). \tag{A.27}$$

If the template $\overline{\mathbf{X}}$ is contained in the W-space, then the state (previously described by equation 6.56) can be rewritten as

$$\mathbf{X} = x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3 + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \tag{A.28}$$

Because the centred second order moments are translationally invariant, and second order homogeneous equations, the following is true

$$\begin{aligned} c'_{xx} &= c_{xx}\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3\right) \\ &= c_{xx}\left(x_1\mathbf{q}_1 + x_1y_2\mathbf{q}_2 + x_1y_3\mathbf{q}_3\right) \\ &= x_1^2 c_{xx}\left(\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right) \end{aligned} \tag{A.29}$$

$$\begin{aligned} c'_{xy} &= c_{xy}\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3\right) \\ &= c_{xy}\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3\right) \\ &= x_1^2 c_{xy}\left(\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right) \end{aligned} \tag{A.30}$$

$$\begin{aligned} c'_{yy} &= c_{yy}\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3\right) \\ &= c_{yy}\left(x_1\mathbf{q}_1 + x_2\mathbf{q}_2 + x_3\mathbf{q}_3\right) \\ &= x_1^2 c_{yy}\left(\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right) \end{aligned} \tag{A.31}$$

where $y_1 \triangleq x_2/x_1$ and $y_3 \triangleq x_3/x_1$. The x_1 term can be eliminated by taking the ratios of c'_{xx}, c'_{xy} and c'_{yy}, giving

$$\begin{aligned} \frac{c'_{xy}}{c'_{xx}} &= \frac{x_1^2 c_{xy}\left(1\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right)}{x_1^2 c_{xx}\left(1\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right)} \\ &= \frac{c_{xy}\left(1\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right)}{c_{xx}\left(1\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right)} \end{aligned} \tag{A.32}$$

$$\begin{aligned} \frac{c'_{yy}}{c'_{xx}} &= \frac{x_1^2 c_{yy}\left(1\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right)}{x_1^2 c_{xx}\left(1\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right)} \\ &= \frac{c_{yy}\left(1\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right)}{c_{xx}\left(1\mathbf{q}_1 + y_2\mathbf{q}_2 + y_3\mathbf{q}_3\right)} \end{aligned} \tag{A.33}$$

The 3 d.o.f. system has now been reduced to 2 d.o.f. in the case that $c_{xx} \neq 0$. This is a fair assumption in practice since $c_{xx} = 0$ implies that the shape has no extent in the x direction. Given solutions for y_2 and y_3, it is possible to solve for x_1 in equation A.30. This immediately gives solutions for x_2 and x_3. The translational degrees of freedom can then be recovered as in equation 6.60.

In the unusual circumstance that the W matrix defining the shape space has the form

$$W = \left(\begin{pmatrix} \mathbf{1} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \end{pmatrix}, \begin{pmatrix} \mathbf{q_1} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{q_2} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ \mathbf{q_3} \end{pmatrix}, \right), \tag{A.34}$$

and the template is contained within the W space, it is possible to reduce the three d.o.f. system to a one dimensional system. This is because the equations relating the control points to the moments have a further structure of homogeneity such that

$$c_{xx} \left(\begin{pmatrix} a\mathbf{x} \\ b\mathbf{y} \end{pmatrix} \right) = a^2 c_{xx} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \tag{A.35}$$

$$c_{xy} \left(\begin{pmatrix} a\mathbf{x} \\ b\mathbf{y} \end{pmatrix} \right) = abc_{xy} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \tag{A.36}$$

$$c_{yy} \left(\begin{pmatrix} a\mathbf{x} \\ b\mathbf{y} \end{pmatrix} \right) = b^2 c_{yy} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}. \tag{A.37}$$

In a method similar to the two dimensional case, a and b can be recovered by solving

the following equation for $t \in [0,1]$,

$$\begin{aligned}
\frac{(c'_{xy})^2}{c'_{xx}c'_{yy}} &= \frac{c_{xy}\left(W\begin{pmatrix}0\\0\\at\\a(1-t)\\b\end{pmatrix}\right)^2}{c_{yy}\left(W\begin{pmatrix}0\\0\\at\\a(1-t)\\b\end{pmatrix}\right)c_{xx}\left(W\begin{pmatrix}0\\0\\at\\a(1-t)\\b\end{pmatrix}\right)} \\
&= \frac{a^2b^2c_{xy}\left(W\begin{pmatrix}0\\0\\t\\1-t\\1\end{pmatrix}\right)^2}{a^2b^2c_{yy}\left(W\begin{pmatrix}0\\0\\t\\1-t\\1\end{pmatrix}\right)c_{xx}\left(W\begin{pmatrix}0\\0\\t\\1-t\\1\end{pmatrix}\right)}. \qquad \text{(A.38)}
\end{aligned}$$

It is then possible to recover a and b by substituting t into the following the pair of equations:

$$c'_{xx} = a^2 c_{xx}\left(W\begin{pmatrix}0 & 0 & t & 1-t & 1\end{pmatrix}^T\right) \qquad \text{(A.39)}$$

$$c'_{yy} = b^2 c_{yy}\left(W\begin{pmatrix}0 & 0 & t & 1-t & 1\end{pmatrix}^T\right). \qquad \text{(A.40)}$$

One question that must be raised is whether it is easier to solve the one or two or the 3 d.o.f. case. One benefit of solving the lower dimensional system is that it is possible to densely (100×100) sample the 2 d.o.f. case in under a second on a standard workstation. On the other hand, if Broyden's method reliably converges in the 3 d.o.f. case, there is no reason to waste the computational cycles sampling. This is a topic that must be further investigated.

Bibliography

Abdallah, S., Nebot, E., and Rye, D. (1994). 2-d object recognition via zernike moments. In *Proceedings, International Conference on Mechatronics and Machine Vision in Practice*, Champain, Illinois. IEEE Computer Society Press.

Abu-Mostafa, Y. and Psaltis, D. (1985). Image normalization by complex moments. *IEEE Trans. Pattern Analysis and Machine Intell.*, 7(1):46–55.

Ahmad, T., Taylor, C., Lanitis, A., and Cootes, T. (1995). Tracking and recognising hand gestures using statistical shape models. In Pycock, D., editor, *British Machine Vision Conference 1995*, volume 2, pages 403–412. British Machine Vision Association, British Machine Vision Association.

Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1987). Active vision. In *Proc. 1st Int. Conf. on Computer Vision*, pages 35–54.

Bangham, A. (1993). Properties of a series of nested median filters, namely the data sieve. *IEEE Transactions on Signal Processing*, 41(1). Reference about median and mode filters.

Bar-Shalom, Y. and Fortmann, T. (1988). *Tracking and Data Association*. Academic Press.

Barnett, S. (1990). *Matrices: Methods and Applications*. Oxford University Press, Oxford, England.

Bartels, R., Beatty, J., and Barsky, B. (1987). *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann.

Baumberg, A. and Hogg, D. (1995a). An adaptive eigenshape model. *Proc. British Machine Vision Conf.*, pages 87–96.

Baumberg, A. and Hogg, D. (1995b). Generating spatiotemporal models from examples. In *Proc. British Machine Vision Conf.*, volume 2, pages 413–422.

Blake, A. and Cipolla, R. (1990). Robust estimation of surface curvature from deformation of apparent contours. In Faugeras, O., editor, *Proc. 1st European Conf. Computer Vision*, pages 465–474. Springer–Verlag.

Blake, A., Curwen, R., and Zisserman, A. (1993a). Affine-invariant contour tracking with automatic control of spatiotemporal scale. In *Proc. 4th Int. Conf. on Computer Vision*, pages 66–75.

Blake, A., Curwen, R., and Zisserman, A. (1993b). A framework for spatio-temporal control in the tracking of visual contours. *Int. J. Computer Vision*, 11(2):127–145.

Blake, A. and Isard, M. (1994). 3D position, attitude and shape input using video tracking of hands and lips. In *Proc. Siggraph*, pages 185–192. ACM.

Blake, A., Isard, M., and Reynard, D. (1994). Learning to track curves in motion. In *Proc. IEEE Int. Conf. Decision Theory and Control*, pages 3788–3793.

Blake, A., Isard, M., and Reynard, D. (1995). Learning to track the visual motion of contours. *J. Artificial Intelligence*, 78:101–134.

Blake, A. and Marchant, J. (1992). Visual sensing and scene interpretation for a precision crop-protection robot.

Blake, A. and Marinos, C. (1989). Shape from texture: estimation, isotropy and moments. *J. Artificial Intelligence*, in press.

Bolle, R., Connell, J., Haas, N., Mohan, R., and Taubin, G. (1996). Veggievision: A produce recognition system. In *WACV96*, page xx.

Brice, C. and Fennema, C. (1970). Scene analysis using regions. *Artificial Intelligence*, 1(3):205–226.

Bucy, R. (1969). Bayes theorem and digital realizations for non-linear filters. *J. Astronautical Sciences*, 17(2):80–94.

Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):679–698.

Cerveron, V., Ferri, F., Domingo, J., Arnau, V., Albert, J., Vicens, M., Pelechano, J., and Ferrer, C. (1990). Un sistem de visión para la recolección automatizada de cítricos. In *II Congr. Iberoamericano de Intelegencia Artificial*, pages 301–316, Morelia.

Cham, T.-J. and Cipolla, R. (1996). Automated b-spline curve representation with mdl-based active contours. In Fisher, R. and Trucco, E., editors, *British Machine Vision Conference 1996*, volume 1, pages 363–372. British Machine Vision Association, British Machine Vision Association.

Cohn, S. and Todling, R. (1995). Approximate kalman filters for unstable dynamics. In *Proceedings, Internation Symposium on Assimilation of Observations in Meteorology and Oceanography*, Tokyo, Japan.

Cootes, T. and Taylor, C. (1992). Active shape models. In *Proc. British Machine Vision Conf.*, pages 265–275.

Cootes, T., Taylor, C., Cooper, D., and Graham, J. (1995). Active shape models—their training and application. *Computer Vision and Image Understanding*, 61(1):38–59.

Cootes, T., Taylor, C., Lanitis, A., Cooper, D., and Graham, J. (1993). Building and using flexible models incorporating grey-level information. In *Proc. 4th Int. Conf. on Computer Vision*, pages 242–246.

Crisman, J. D. (1992). Color region tracking for vehicle guidance. In Blake, A. and Yuille, A., editors, *Active Vision*, chapter 7. The MIT Press.

Curwen, R. (1993). *Dynamic and Adaptive Contours*. PhD thesis, University of Oxford.

Dalton, B., Kaucic, R., and Blake, A. (1995). Automatic speechreading using dynamic contours. In *Proceedings NATO ASI Conference on Speechreading by Man and Machine: Models, Systems and Applications*, pages 373–382. NATO Scientific Affairs Division.

Dickmanns, E. and Graefe, V. (1988). Applications of dynamic monocular machine vision. *Machine Vision and Applications*, 1:241–261.

Diercks, P. (1993). *Curve and Surface Fitting with Splines*. Clarendon Press.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. John Wiley and Sons.

Durrant-Whyte, H. (1988). Uncertain geometry in robotics. *IEEE J. of Robotics and Automation*, 4(1):23–31.

Faugeras, O. (1993). *3D Computer Vision*. MIT Press.

Ferrier, N., Rowe, S., and Blake, A. (1994). Real-time traffic monitoring. In *Proc. 2nd IEEE Workshop on Applications of Computer Vision*, pages 81–88. IEEE.

Fleck, M., Forsyth, D., and Bregler, C. (1996). Finding naked people. In *Proc. 4th European Conf. Computer Vision*, pages 593–602, Cambridge, England.

Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1990). *Computer Graphics: Principles and Practice*, chapter 13, pages 563–604. Addison-Wesley.

Geman, D. (1987). A stochastic model for boundary detection. *J. Image and Vision Computing*, 5:61–65.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Gennery, D. (1992). Visual tracking of known three-dimensional objects. *Int. J. Computer Vision*, 7(3):243–270.

Gonzales, R. and Woods, R. E. (1992). *Digital Image Processing*. Addison-Wesley.

Goodwin, C. and Sin, K. (1984). *Adaptive filtering prediction and control*. Prentice-Hall.

Grewal, M. and Andrews, A. (1993). *Kalman Filtering: Theory and Practice*. Prentice Hall, Englewood Cliffs, New Jersey.

Harrel, R., Hood, C., Moltó, E., Munilla, R., Bieniek, M., and Cantliffe, D. (1993). Machine vision based analysis and harvest of somatic embryos. *Computers and Electronics in Agriculture*, 9:13–24.

Harris, C. (1992a). Geometry from visual motion. In Blake, A. and Yuille, A., editors, *Active Vision*, pages 263–284. MIT.

Harris, C. (1992b). Tracking with rigid models. In Blake, A. and Yuille, A., editors, *Active Vision*, pages 59–74. MIT.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151.

Hogg, R. and Tanis, E. (1988). *Probability and Statistical Inference*, chapter 2. Macmillian Publishing Company, third edition.

Hopcroft, J., Huttenlocher, D., and Wayner, P. (1992). *Affine Invariants for Model-Based Recognition*, pages 354–374. The MIT Press.

Horn, B. and Schunk, B. (1981). Determining optical flow. *J. Artificial Intelligence*, 17:185–203.

Huttenlocher, D., Noh, J., and Rucklidge, W. (1993). Tracking non-rigid objects in complex scenes. In *Proc. 4th Int. Conf. on Computer Vision*, pages 93–101.

Isard, M. and Blake, A. (1996). Visual tracking by stochastic propagation of conditional density. In *Proc. 4th European Conf. Computer Vision*, pages 343–356, Cambridge, England.

Kapur, J., Sahoo, P., and Wong, A. (1985). A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics, and Image Processing*, 29(3):273–285.

Kass, M., Witkin, A., and Terzopoulos, D. (1987). Snakes: Active contour models. In *Proc. 1st Int. Conf. on Computer Vision*, pages 259–268.

Kaucic, R., Dalton, B., and Blake, A. (1996). Real-time liptracking for audio-visual speech recognition applications. In *Proc. 4th European Conf. Computer Vision*, pages 376–387, Cambridge, England.

Klinker, G., Shafer, S., and Kanade, T. (1988). The measurement of highlights in color images. *Int. J. Computer Vision*, 2:7–32.

Koenderink, J. (1986). Optic flow. *Vision Research*, 26(1):161–179.

Koller, D., Weber, J., and Malik, J. (1994). Robust multiple car tracking with occlusion reasoning. In *Proc. 3rd European Conf. Computer Vision*, pages 189 – 196. Springer-Verlag.

Langford, M. (1984). *The Darkroom Handbook*, chapter 10, pages 241–2. Focal Press, 5th ed. edition.

Lanitis, A., Taylor, C., and Cootes, T. (1995). A unified approach to coding and interpreting face images. In *Proc. 5th Int. Conf. on Computer Vision*, pages 368–373.

Lefebvre, M., Gil, S., Brunet, D., Natonek, E., Baur, C., Gugerli, P., and Pun, T. (1993). Computer vision and agricultural robotics for disease control: The potato operation. *Computers and Electronics in Agriculture*, 9:85–102.

Lowe, D. (1992). Robust model-based motion tracking through the integration of search and estimation. *Int. J. Computer Vision*, 8(2):113–122.

Marchant, J. (1991). Intelligent machinery for agriculture. *Machine Vision and Applications*, 151:177–186.

Marchant, J. and Brivot, R. (1995). Real time tracking of plant rows using a hough transform. *Real Time Imaging*.

Marr, D. and Hildreth, E. (1980). Theory of edge detection. *Proc. Roy. Soc. London. B.*, 207:187–217.

Maybank, S., Worrall, A., and Sullivan, G. (1996). Filter for car tracking based on acceleration and steering angle. In *Proc. British Machine Vision Conference*, Edinburgh, Scotland.

Modayur, B., Shapiro, L., and Haralick, R. (1992). Visual inspection of machined parts. In *Proc. Conf. Computer Vision and Pattern Recognition*, Champain, Illinois.

Moltó, E., Plá, F., and Juste, F. (1992). Vision systems for the location of citrus fruit in a tree canopy. *J. Agric Eng. Res.*, 52:101–110.

Moses, Y., Reynard, D., and Blake, A. (1995). Determining facial expressions in real-time. In *Proc. 5th Int. Conf. on Computer Vision*, pages 296–301, Boston.

Mundy, J. and Zisserman, A. (1992). *Geometric invariance in computer vision*. MIT Press.

Murray, D. W., Bradshaw, K. K., McLauchlan, P. F., Reid, I. D., and Sharkey, P. M. (1994). Driving saccade to pursuit using image motion. *IJCV*. To Be Published.

Nalwa, V. (1993). *A Guided Tour of Computer Vision*, chapter 1, pages 3–25. Addison-Wesley.

Nevatia, R. (1977). A color edge detector and its use in scene segmentation. *IEEE Transactions on Systems, Man and Cybernetics*, 7(11):820–826.

Nevatia, R. (1982). *Machine Perception*, chapter 6, pages 90–127. Prentice Hall.

Pentland, A. and Horowitz, B. (1991). Recovery of nonrigid motion and structure. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7:730–742.

Plá, F., Juste, F., Ferri, F., and Vicens, M. (1993). Colour segmentation based on a light reflection model to locate citrus fruits for robotic harvesting. *Computers and Electronics in Agriculture*, 9(1):53–70.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1988). *Numerical Recipes in C*. Cambridge University Press.

Prokop, R. and Reeves, A. (1992). A survey of moment-based techniques for unoccluded object representation and recognition. *Computer Vision, Graphics and Image Processing*, 54(5):438–460.

Rabatel, G. (1994). Harvesting the fruit of vision. *Image Processing*, 6(3):16–20.

Ramachandran, V. (1990). Visual perception in people and machines. In Blake, A. and Troscianko, T., editors, *AI and the Eye*, chapter 3, pages 21–78. John Wiley and Sons.

Reynard, D. (1993). *Nonlinear Estimation*. PhD thesis, University of Newcastle Upon Tyne.

Reynard, D., Wildenberg, A., Blake, A., and Marchant, J. (1996). Learning dynamics of complex motions from image sequences. In *Proc. 4th European Conf. Computer Vision*, pages 357–368, Cambridge, England.

Rothwell, C., Zisserman, A., Mundy, J., and Forsyth, D. (1992). Efficient model library access by projectively invariant indexing functions. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 109–114.

Rowe, S. (1996). *Robust feature search for active tracking.* PhD thesis, University of Oxford.

Rowe, S. and Blake, A. (1995). Statistical background modelling for tracking with a virtual camera. In *Proc. British Machine Vision Conf.*, volume 2, pages 423–432.

Sato, J. and Cipolla, R. (1994). Extracting the affine transformation from texture moments. Technical Report CUED/F-INFENG/TR 167, University of Cambridge.

Shafer, S. (1985). Using color to separate reflection components. *COLOR Res. Appl.*, 10:210–218.

Shapiro, L. (1993). *Affine Analysis of Image Sequences.* PhD thesis, The University of Oxford.

Shapiro, L., Wang, H., and Brady, J. (1992). A matching and tracking strategy for independently moving objects. In *Proc. British Machine Vision Conf.*

Singh, N., Delwiche, M., and Johnson, R. (1993). Image analysis methods for real-time color grading of stonefruit. *Computers and Electronics in Agriculture*, 9:71–84.

Stephens, R. (1990). Real-time 3D object tracking. *J. Image and Vision Computing*, 8(1):91–96.

Strang, G. (1976). *Linear Algebra and Its Applications.* Harcourt Brace Jovanovich, San Diego.

Taylor, M., Blake, A., and Cox, A. (1994). Visually guided grasping in 3D. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 761–766.

Teh, C. and Chin, R. (1988). On image analysis by the method of moments. *IEEE Trans. Pattern Analysis and Machine Intell.*, 10(4):496–513.

Terzopoulos, D. and Szeliski, R. (1992). Tracking with Kalman snakes. In Blake, A. and Yuille, A., editors, *Active Vision*, pages 3–20. MIT.

Tominaga, S. (1990). A color classification method for color images using a uniform color space. *Proc. Conf. Computer Vision and Pattern Recognition*, pages 803–807.

Trevelyan, J. (1992). *Robots for Shearing Sheep – Shear Magic.* Oxford University Press.

Wang, H. and Brady, J. (1992). Vision for mobile robots. *Proc. Roy. Soc. Lond. B.*, 337.

Yow, K. and Cipolla, R. (1995). Scale and orientation invariance in human face detection. In *Proc. British Machine Vision Conference*, Birmingham, England.

Zisserman A., Forsyth D., Mundy J., Rothwell C., Liu J., and Pillow N. (1995). 3D object recognition using invariance. *Artificial Intelligence*, 78:239–288.

Zucker, S. W. (1976). Region growing: Childhood and adolescence. *Computer Graphics and Image Processing*, 5(3):382–399.